COSC345 Software Engineering

CommunAphasia:

Alpha Release Documentation

RedSQ Team:

Winston Downes	(2872112)
Max Huang	(4427762)
Mitchie Maluschnig	(9692597)
Sam Paterson	(3175949)

28 May 2018

How to use the app

Build project on iPad Pro (10.5 inch) simulator (we haven't got screen sizing working perfectly yet). From the main menu you will have two options: ‘Image to text’ or ‘Text to image’.

Image to text – this is significantly more complicated so needs a bit more work. The user can pick an image (or multiple) from the selection, choose whether the word is singular or plural, and the image will then be placed in the bottom area to show which images you have selected. Once you are happy with your “sentence”, press done and you will be greeted with your sentence as words which includes some processing (adding appropriate function words and making it more coherent) you will also see your original image selection at the bottom of the screen.

Text to image – the user inputs a sentence and presses done. You will then be greeted with a screen that shows the image results and also the sentence you typed, at the bottom of the screen

Pressing ‘back’ in both cases will move back a screen and let you re-input a different “sentence” from scratch.

How it works

Text-to-image

The user types in a sentence they want to construct, the text is parsed to our algorithm and the corresponding images are pulled out of the database and displayed on screen.
If the user enters a sentence containing words (which describe images) which are not available in our database, the user will be asked to change those words.
Currently, we do not have a substantial amount of images in our database so only a limited amount of sentences can be made. The following sentences are some of the examples of ‘translation’: (words-to-images)

- The blue fish is sleeping
- I am eating dinner
- The big man is calling the small woman

These sentences can also be translated be into words by pressing the relevant pictures in the order that one would speak the words. All the available words to be used to construct sentences (at this stage) are contained in a CSV file “images.txt”, using the first

element of each line. Since adding more images is simply content generation, it should be more comprehensive by the beta release.

Image-to-text

The user is presented with a default set of images with which they can use to make sentences from. (Following this alpha, there will be more images added and also categories of images which can be brought up on demand).

The user constructs a sentence by tapping on the images they want (in order) and those are placed in the field at the bottom of the screen. Once the user is finished, they tap the ‘done’ button and the images are passed into our image-to-text algorithm which identifies the parts of speech for that word (e.g. noun) and creates a coherent sentence by padding the sentence out with the appropriate function words depending on the current picture and the surrounding ones. For example, if the current picture is a cat(noun), the definite article ‘the’ will be placed ahead of the word ‘cat’. If the next image is the picture representing eating, then the third-person verb to be (is) will be placed next in the string, with the word cat after it.

Issues encountered

To begin with, Sam and Max had previous experience with Sourcetree and git so they taught it to the rest of the team, in depth - with emphasis on strict procedures of how to push and pull - we agreed that only Max and Sam would handle merges at this stage until the rest of the team were more familiar and comfortable with version control. We all branched out once the project was created and worked on it separately, we soon found out that Xcode doesn’t play well with Sourcetree and encountered an array of issues when dealing with source control- so we decided to move to Xcode’s version control. This change meant we wasted all that time training the team on Sourcetree and we are now using a completely new system we have to all learn which has inevitably led to problems and wasted time.

We were originally using git with GitLab for source control but we found that it had sporadic outages, that coupled with major Xcode merge issues pushed us to explore other options. No one really knew how Xcode worked so it was hard to troubleshoot what was going wrong. The choice to move our repo was not a light one, Max worked with Mitchie to solve the push, pull, and merge Mitchie’s branch to master but there was an issue with a missing framework for SQLite. This issue took us days to fix, the result was that no one could work on the project as everything was dependent on the SQLite code (and framework)

to provide the required backbone on which to build the rest of the project. It would exist on the local machine but not when it was merged into another branch. Due to all of this, we made a consensus to make a fresh repo and only pull in one branch of work from the old repo. We decided to go with BitBucket as it allowed us to create a private repo at no cost (in comparison to GitHub).

We backended Xcode onto BitBucket and and used Xcode's built-in source control features with git. During development, we came across quite a few issues with merging, specifically error messages like "xcode fatal: cannot do a partial commit during a merge" and Xcode "tree conflicts". Xcode refused to resolve these issues and we tried all the built-in tools to do so. Max has previous git experience and a good handle with terminal and was able to sort out these issues - the partial commit issue didn't allow us to commit anything. All the conflicts were resolved but still wouldn't allow us to commit. This was solved by removing the "MERGE_HEAD" and then the commit would go through without a hitch. The second issue with the "tree conflicts" didn't allow us to merge, at all. Max resorted back to using SourceTree and merging the two branches manually, once that was done, it was committed via Xcode and the issue was resolved.

Regarding linguistic issues, we found that there are a lot of words that cannot not be well represented with pictures. An example of this are the modal verbs, such as want, need, should etc. These words are very common but are not easy to portray with a picture. One way that we are overcoming this issue is expanding the learning curve that the patient will have to surmount when just starting to use this app and having the patient rote learn particular buttons that represent these words. As these words are not symbolically amenable to representation, this will require some training, because there will be no intuitive link between the picture, or symbol representing the modal verb and that verb. As there are only a limited number of these particular verbs, and because they are used often, it is expected that the patient will eventually learn these.