Documentation for future enhancements

This guide is an introduction to our deliverable code, intended to assist anyone working on our codebase in the future.

0.1 Overview

Our deliverable is a prototype web application, built on top of Node.js. Node is an event-driven framework for server-side JavaScript, based on Google's V8 engine. It is open-source and under active development; in addition, it has enjoyed great popularity recently and there is thus no shortage of libraries and learning resources. More information can be found at `http://nodejs.org/`.

We make use of MongoDB, a document database, to store persistent data, such as the attribute vectors for images and destinations. MongoDB is not related to SQL; it stores untyped 'documents' in 'collections' (compare SQL: typed 'rows' in 'tables'). Despite being untyped, MongoDB is very high-performance, and because the document syntax is JSON, it's easy to store and retrieve JavaScript objects. More information about MongoDB can be found at `https://www.mongodb.org/`.

The client-side is implemented in jQuery. JQuery is a JavaScript library that embeds a functional domain-specific language for document traversal, manipulation, event-handling, AJAX, and more. More information here: `http://jquery.com/`.

0.2 Dependencies

In addition to Node.js and MongoDB, the following Node.js libraries are required for operation. See the Installation section for instructions on how to install them.

- Express (`http://expressjs.com/`): lightweight web application framework
- Mongoose (`http://mongoosejs.com/`): MongoDB interface with schema-based object modeling
- Cheerio (`https://github.com/MatthewMueller/cheerio`): server-side jQuery implementation
- Request (`https://github.com/mikeal/request`): simple HTTP client

0.3 Installation

First, install Node.js and MongoDB. Steps for this will depend on your platform; see the respective websites for detailed instructions.

Once this is done, clone the repository. At the time of writing, the repository is available at

```
https://github.com/vmohapatra/snow-white
```

however, this will likely change in the future.

Next, run `npm install -d` from within the project directory. This will install the necessary dependencies listed in `package.json`.

Finally, copy the images from a remote location to

```
<project folder>/img/images/
```

or create a symbolic link from that point to the image directory. The images are not included in Git due to size concerns.

0.4 Running

Once installed, the server can be started by running `node server.js 80`. 80 is the port number; any other value can be used here, but 80 is the default for HTTP and the port that browsers will try automatically if none is given in the URL.

0.5 Components

There are three main components of the application: the client, the server interface, and the backend.

`public/js/client.js` is the client JavaScript code. It is embedded inside of `public/main.html` and run by the user's browser. The client code

handles click events from the browser, makes the appropriate AJAX calls, and renders the results of those calls back into the document.

`server.js` is the server interface. It's fairly succinct, largely thanks to the Express framework. It is responsible for serving the `public/main.html` page, and for handling AJAX requests (through the `/live` endpoint). Additionally, `server.js` calls into `auth.js` in order to authenticate users against the database, and into `backend.js` to service AJAX requests.

`backend.js` is the backend. It does the heavy-lifting of deciding what images and destinations to return to the client. There are two main functions in `backend.js`: `backend.images` and `backend.dests`. The former takes a `user` object and a `game` object, both retrieved by `auth.js`, as well as a number; it calculates the cosine-similarity for the amalgamated tag vector of selected images and each unseen image, and selects images to return with probability proportional to the resulting dot product (see Algorithms section above for more details). The latter function does more or less the same thing for destinations, with two changes: first, the highest rated destinations are returned, not a weighted random selection, and second, images with known locations are also used as recommended destinations. After computing a set of destinations, `backend.dests` requests Wikivoyage.org pages for each and extracts the first portion of the article for use as a short description of the destination.

The rest of the files in the repository are support files that perform various minor functions. Specifically:

- `auth.js`: authenticates users against MongoDB
- `config.js`: reads command-line args and sets server options
- `db.js`: connects to MongoDB instance and imports models
- `util.js`: contains several utility functions used elsewhere
- `models/*.js`: models of MongoDB document types for Mongoose.