1 Introduction

This project is one encompassing natural language processing (nlp) and full stack web development. Regarding nlp, we are in the realm of emotion analysis as we are seeking to algorithmically discern emotion within text. We largely take our conceptions of emotion from Robert Plutchik:

!plutchiks wheel [?Plutchik7:online]

> Emotion is a complex chain of loosely connected events that begins with stimulus and includes feelings, psychological changes, impulses to action and specific, goal-directed behavior. [?10.2307/27857503]

He provides a wheel of emotions related to color theory, such that an emotion can be seen as a mixture of emotions (i.e. "anticipation" lays between "joy" and "anger"). Emotion detection within text is a rather difficult problem, although it has grown in popularity recently. As it is an emerging field, there is a lack of available datasets, as described by Seyeditabari et al.. They mention several available datasets, as well as different methods in use, including the affect dataset from Mohammad and Turney [?Mohammad13] that we use. The methods noted fall into the supervised and unsupervised categories: the supervised methods include data collection through Twitter scraping and large surveys, as well as their use within support vector machines (SVM), naive Bayes classifiers, k-nearest neighbor (KNN) clustering, and more; the unsupervised methods include Probabilistic Latent Semantic Analysis (PSLA), simple lexical approaches, and point-wise mutual information (PMI) [`?DBLP:journals/corr/abs-1806-00674`]. We will make two unsupervised models for emotion detection based on a subset of emotions from Plutchik's wheel (anger, sadness, joy, and fear); the first model will take a lexical approach, where we take the average token scores for each affect based on Mohammad and Turney's word-affect and affect-intensity lexicons; the second will be based on latent Dirichlet allocation (LDA), a generalization of PSLA by Blei et al. [`?david\textit{}m.\textit{}y.\textit{}i.\textit{}1970`]. In addition to our two emotion-detection models, we look into word replacement models based on a thesaurus [`?BigHugeT88:online`] and our scoring models. One of these models performs arbitrary replacement based on synonyms of tokens; the other seeks to maximize score in one of our affect categories.

A large focus was placed on full-stack web development: dataset creation/consumption, database management, server configuration, creating REST APIs, and creating a web frontend. We leverage several technologies for this, including Python with Flask[`?Welcomet33:online`], Golang and the mongo-go-driver[`?mongogod83:online`], HTML/CSS/Javascript via VueJS[`?Vuejs17:online`], and Docker[`?Empoweri41:online`].

!architecture diagram

This diagram serves to describe our application. A user requests our site, and our reverse-proxy forwards this request to our frontend server. Then, when the user accesses the thesaurus page, they see results from calling our CRUD-wrapper API that accesses our database. When they use the models on the site, they are making requests to the model API—which then makes requests to the CRUD API. For scalability, ease of deployment, and security benefits, the frontend, model API, CRUD API, and database are all in containers. All of this sits on a virtual machine from DigitalOcean[`?Digital01:online`], with the code available on Github[`?GitHub86:online`].

We will begin discussing server configuration, move to database construction, then our CRUD API, followed by our models, the frontend, and finishing with results.