CS410 Project Presentation: Food Recipe Search Engine

Team name: F20_TIS_DEV_TEAM

Team Members:

Jon LaFlamme (captain)	jml11@illinois.edu
Pradeep Sakhamoori	ps44@illinois.edu
Rahul Sharma	rahul9@illinois.edu

Project Overview: This application is a vertical search engine for food recipes. Special features include auto-complete and auto-suggestion in the search bar achieved with character and word n-grams. Search results are rendered on the same page as the search bar. Our system is configured with and run on Elasticsearch. Included with the project but not integrated with the search application is a content-based recommender system.

Project Review Roadmap

Recommended

Project overview video:
https://youtu.be/8MrQOFWIooo

Installation instructions:
README.md

Supplementary

Video install guides in *README.md*

Recommender system video demo:
https://youtu.be/pEuqzOdScEQ

Recommender system writeup:
Recommender_System.pdf

Search engine and project architecture:
https://youtu.be/8rgRQOhtluQ

Special Consideration: Our team originally consisted of four members through the formation, proposal, development and progress report phases of the project, but one of our team members dropped this course prior to providing assigned deliverables. Per the TA, we were instructed to notify our reviewers to take this development into consideration in the evaluation of our final submission, as it has necessitated some adjustments to the scope of this project.

Application/Software Functionality: This application is a food recipe search engine with a dataset of over 65,000 food recipes. Word and character-based autosuggest and autocomplete are supported dynamically in the search bar. Search results are dynamically displayed on the same page as the search bar, with matched query terms

highlighted in the title field of each recipe that is rendered in the search results. Additionally, a content-based recommender system was developed and validated on the author's local machine. It is not integrated with our search application and is thus considered an 'extra' in terms of our project submission. To review the recommender module, simply watch the demo video and read the pdf.

Application/Software Implementation:

Software architecture:

- batch -> data ingestion modules; recommender module
- dataset -> food recipe datasets in JSON format
- es-setup -> Elasticsearch configuration files
- search-engine-webapp ->
  - app -> application driver and API, browser files and templates
  - search_engine_egg.info -> Application setup files
  - user-profile -> our beta-recommender with user-profile as JSON
- AUTHORS.txt -> source code attributions
- README.md -> Installation instructions
- Dockerfile.amd -> Docker build script
- pdfs (proposal, progress report, final documentation, recommender system)

Description of Team Member Contributions:

All members: Weekly team meetings, topic exploration, identification of datasets, learning Elasticsearch and Kibana, shared contributions to proposal, progress report and final submission documents.

Jon: Administrative tasks of coordinating, authoring or editing and submitting project deliverables. Assisted with troubleshooting and documenting Docker installation requirements on Mac. Spent several hours familiarizing myself Elasticsearch Assisted with automation of Elasticsearch data ingestion with data_loader.py. Assisted with user interface development (parsing JSON search results, and displaying search results as hyperlink recipe titles on same page as search bar). Assisted with bug fixes recommender API. Generated ~1500 test queries by collecting most popular 'recipe' queries using Google Trends (not implemented in final submission). Developed algorithm for sorting recipes search results based on complexity/difficulty (not implemented in final submission). Developed content-based recommender system (included in project but not integrated with our final application.)

Pradeep: Started with spending time brainstorming project architecture, design discussions, planning, task allocation, internal milestones and deadlines (We used Microsoft teams to collaborate, drive weekly tasks and discussions). Initially started with contributions to look for available open source food recipe datasets, found couple of them which are publicly available (from Kaggle) and analyzing its format and its usability. We used Elasticsearch (ES) engine, python and

flask interface for programming – Initial challenge is to get the environment ready to unblock others with a base line application (by pulling initially committed Skelton changes from team). Then spent time containerizing the complete application. Spent time understanding on launching elastic server as backend process from Docker with Ubuntu 16.04 environment – figuring out missing components in terms of launching web app and tunneling ES web server to localhost machine web-server – Debugging, fixing bug and released initial docker containerized app which was quite challenging. Developed/Coded basic/ initial pipeline for recommender system (contextual based) by pre-populating UserProfile (static) with specific food interest and which will be used for querying from ES(which is current a standalone module – which user can trigger from docker bash). Spent time supporting team with guidance on fixing docker issues on Mac. Rest of the time spent in testing application on Ubuntu.

Rahul: Assisted with identification of dataset, set up project collaboration space on Trello. Built project architecture template for search application with Flask. Designed and authored Elasticsearch index configuration and ingestion scripts. Authored setup instructions and scripts for application build without Docker. Designed and authored user interface for search application. Designed and authored autosuggest and autocorrect features with Elasticsearch and Flask.

Methodology and analysis: Please see included supplementary videos and documents for a more in-depth coverage of design choices, implementation details and results analysis.