CS410 Text Info System - Course Project Documentation

Fall 2023

Qinxi Wang

qinxiw2@illinois.edu

1) An overview of the function of the code (i.e., what it does and what it can be used for).

We will go over the three main folders: Data, Model, and Search in the code directories, with an overview of the directory's tasks, and annotated functionalities of the subfiles and their functionalities.

/Data directory contains everything exploratory on data analysis, cleanup, and creation.

- cleaned_data.csv.gz
    - the cleaned compressed csv file using data_cleaning.py
- data_cleaning.py
    - This data processing task that provides functionality creating unique IDs for wines, leveling geographic fields, adding currency information to prices, cleaning text columns, extracting wine years, imputing missing data, removing duplicated values, indicating repeated viewers with unique taster-title combinations, identifying and correcting typos in similar wineries, provinces, regions, and designations, and performing NLP cleanup on reviews.
- data_model_exploration.ipynb
    - the early interactive exploration and modeling notebook fil, added it here for reference only and to show our work, and serve as a data lineage and experiments log; the insightful parts of the work are later turned into more rigorous python functions in various directories in this project.
- prime_search_database.py
    - This data processing initiative involves loading a refined dataframe that has undergone both cleaning and sentiment fine-tuning. Following this, the sentiment dictionary column is divided into distinct label and score columns. A cohesive regional level is established across two regions, and for optimization in search functions, pertinent columns are selected. The conclusive dataframe is then saved to an output file.
- search.csv.gz
    - the primed compressed csv file using prime_search_database.py
- sentiment_data.csv.gz
    - the sentiment data analyzed and fine_tuned dataframe into a compressed csv file using Model/sentiment_analysis_and_fine_tuning.py

/Model directory contains everything model experiments, training, and evaluation:

- cf_recommender.py
    - The code here creates a collaborative filtering-based recommender system. First the dataframe from section above is loaded and divided into train and test sets, with the generation of user and item IDs. Subsequently, a tfidf_matrix is built using the training set for wine descriptions. The next step involves creating cosine similarities for each pair of wines and reviews. Further enhancement is achieved by combining additional relevant features with text review data. Finally, the resulting models and artifacts are saved to output files for inference use.
- knn_recommender.py
    - This task creates a recommender system using the k-nearest neighbor model. The process steps similarly by loading the dataframe used above and partitioning it into training and testing sets, alongside the creation of user and item IDs. Then it constructs a data pivot frame and matrix by incorporating information about the wine and sentiment of the review. Then, a k-nearest neighbor model is trained using the data matrix generated earlier. An simple test is introduced for pulling a recommendation for a single wine id input, facilitating queries for random indices and retrieving recommendations. The concluding step involves saving the finalized models and artifacts to output files so it can be used by search later.
- rf_recommender.py
    - This workflow prepares data, trains and evaluate a random forest model for wine recommendation. The initial step involves splitting the data into training and testing sets. Following this, a random forest classifier model is trained using the training data. The subsequent task encompasses the evaluation of the trained model, with metrics including precision, recall, confusion matrix, accuracy, and a comprehensive classification report.
- sentiment_analysis_and_fine_tuning.py
    - This task performs sentiment analysis and fine-tuning for wine reviews in our dataset. It loads the dataframe and prepares the pipeline for a pre-trained model chosen from the available list for sentiment analysis. Then the selected pipeline and model are utilized to run sentiment analysis on the wine reviews. A baseline analyzer output is examined, and fine-tuning is performed by splitting the existing data into training and testing sets, utilizing points as a proxy for sentiment level. Following this, a model is trained on top of the existing one using a trainer and the fine-tuning data. The sentiments are updated by rerunning the final model, and the ultimate output is saved to a CSV file mentioned in /Data section above.

/Search directory contains everything search, retrieval, and model inference along with the artifacts needed:

- matrix/df*.pkl

    - the output from model training earlier, artifacts needed for lookup in inference along with the model

- model*.joblib
    - The output from model training earlier, the model itself needed to run inference

- rec_comment_sim.py
    - This task runs inference for the cf model using tfidf matrix for similar wine and review matching. The main function will be invoked by the rec inference function, and pass in the matched wine attributes from the recs based on search for the user, then recommender then works to find similar wines based on search match and also draw out the bag of words in the review that are shared between the similar wines.

- rec_inference.py
    - This task runs inference for the knn model for similar wine based on wine attributes and sentiments of the review. The main function will be invoked by the search wine function, and pass in the matched wine attributes from the user, then recommender then works to find similar wines based on search match

- search_wine.py
    - This is the main task where a user can search, select,and retrieve similar / recommended wines. The functionality is built using interactive command line input exchange, where a user can search in English words or short phrases what type of wine (e.g 'sherry', 'italian') or wine attributes they want (e.g 'aromatic', 'fruit flavors'). These are then used as search phrases against our database, the top retrieval quality wines are then anchored to use for additional recommendation inference, and added the final results matched for the user: 'we found these wines for you based on your search, similar wines related to your search also include...'

All the functions provided in code here can all be easily transferred and applied on other similar datasets in various domains.

2) Documentation of how the software is implemented with sufficient detail so that others can have a basic understanding of your code for future extension or any further improvement.

We began with data exploration, we performed standard cleaning tasks such as checking duplicates, filling Nan values, and ensuring the data of the same column share the same data types. We also attempted to fill missing values at this stage, such as by extracting wine year from the title, level geographic fields by existing records, and impute price info. The most significant contribution and implementation of our software at this stage is the text processing and cleaning, where we first tokenized the fields with word and sentence tokenizer,

then checked spelling and removed overly long or short words, and stopwords. We then lemmatized the words, and dropped numerical and non-english words from the reviews.

We stretched out search functionality after that. We tried a few libraries for that, weighing search accuracy and retrieval speed. We landed on using fuzz function from rapidfuzz library, which yielded the best performance compared other libraries as well as our inhouse implementation. The search wine part is implemented with user interaction in mind, the program starts by prompting users to search for a wine title, attributes, or other short phrases they'd like to find wine that match the criteria, then search through multiple columns such as reviews, variety, winery, region, etc to find best matching wines. Then from there it will evoke the recommendations software described below, to retrieve similar wines we think the user is going to like based on their search, and other search phrase suggestions if they liked what they see.

After that we implemented the sentiment analysis and fine-tuning task. Per our proposal plan, we surveyed and picked a pre-trained sentiment analyser model most suitable for our dataset, and then applied fine tune and transfer learning on top using the dataset we have. The purpose of sentiment analysis is two folds - one is to understand the opinions included in the wine dataset better, and see if there's correlation between wine attributes and sentiment; the other is we believe sentiments can play a role in text feature based recommendations, so we want to utilize sentiment as a signal to compare different wines for similarities. Since our dataset itself does not have labels suitable for training a sentiment classifier from scratch, we applied fine-tuning with a test and train set we sampled from the dataset and used points as proxy.

By now we had everything we needed to go into recommender system implementation, here the software is implemented by two stages. First stage is offline modeling and evaluation. At this stage all the code was first sandboxed in the data exploration notebook, we implemented various classic recommender systems, including collaborative filtering, discriminative model, instance-based learning algorithm, and text to embedding feature. From there we picked three most promising types of each to prototype - collaborative filtering using SVD, k-nearest neighbor, and random forest. We cleaned up the code and combined training, evaluation, and model and artifacts saving into their corresponding file under Model. The second stage of recommendation is online inference. We have the recommendation model and their artifacts created and stored along in the Search/ dir, where the search match results will invoke these models to be loaded into memory if have not already, and take in the matched wine as input, and perform necessary pre-processing, lookup, similarity calculation for recs, along with search recommendation for refined search or pattern mining from the wine reviews for user discovery.

The last pieces of the software are about user interface, we wired up the properly refactored and cleaned working modules from above, designed dialogues and prompts to keep users engaged. Here is a sample dialogue after a user enters the program:

Let's start by telling me about what wine you are looking for. Type exit to leave anytime.

you can search with a short phrases what type of wine (e.g 'sherry', 'italian') or wine attributes (e.g 'aromatic', 'fruit flavors'): sherry

Got it! Based on sherry we found the following wine title based on your search:

1 : Pedro Romero NV Full, Dry Oloroso Sherry (Jerez) from Spain- $13

2 : Domaine de Savagny 2011 Savagnin (Côtes du Jura) from France- $52

3 : González Byass NV Solera 1847 Cream Sherry (Jerez) from Spain- $26

For other wines similar to your search, we also recommend:

1 : Osborne NV Cream Sherry (Jerez) from Spain- $18

2 : Delgado Zuleta NV Premium Fino Sherry (Jerez) from Spain- $16

3 : Lustau NV Dry Amontillado Los Arcos Sherry (Jerez) from Spain- $17

Here are some wine variety we think you will like: palomino, with recommendation score: 75%

We found these top common words for the matched wine reviews, if you find anything you like you can search more with them:

'brown sugar, stone fruits, dried apricot, caramel toffee, nutty flavors, palate flavors', 'garnacha blanca, fruit flavors, acidity flavors, white garnacha, peach pit, green apple', 'flavors lemon, lemon green, apple peach, flavors finish, tropical fruit'

Any other wine search? Type exit to leave anytime: …

3) Documentation of the usage of the software including either documentation of usages of APIs or detailed instructions on how to install and run a software, whichever is applicable.

For a quick start of the main application software, search and recommend wine, run the following:

```
python3 -m venv myenv  # Python 3.11.4 used by author

source venv/bin/activate

pip install -r requirements.txt

python Search/search_wine.py
```

Always make sure you are in the parent dir level as the README when you run any of the scripts. After the main dialogue program is started, simply follow the prompt to search for wine or reviews, and explore matched results and related recommendations for you!

For all the supplementary software, the setup remains the same: start a python virtual env from the command line, stay in the parent directory level and install libraries included in the requirements.txt. All the .py files are executed, simply swap `search_wine.py` with your choice of the file. E.g directly run `python Search/rec_inference.py` or `python Search/rec_comment_sim.py`

If you are most interested in training and eval a few different recommendation models, sentiment analysis with fine tuning, or experimenting with different model architecture and/or parameters, all Model/ programs are automatically run from start to finish without needing interference, and can be done independently of each other. E.g `python Model/cf_recommender.py, python Model/knn_recommender.py, python Model/rf_recommender.py`. Note that these files use the output from Data/ by default, if you would like to swap in a file of your choice, simply override the pd.read_csv() path to be your local file.

If you are supplying your own dataset but want to leverage the cleaning process run the cleaning scripts under Data/ and follow the input to specify your own data location. E.g `Data/data_cleaning.py --file_location_path {your_file_loc}`

All the scripts are annotated with docstrings on top of the file. You can also watch the demo and/or slides supplementary with this project in the repo, which includes more detailed walkthrough of each component, and contains additional screenshots and output samples.

4) Brief description of contribution of each team member in case of a multi-person team.

I am the only member of the team. All the tasks described here are completed by me fully.