The Yelper Helper:
Providing recommendations to Yelp users based on their prior review history

Team Members: Aditi Goyal (adigoyal), Akayla Hackson (akayla), Neeraj Mathur (mathurn)

1. Introduction

Yelp is a widely used platform designed to recommend businesses and services. Despite its popularity, the platform can be cumbersome to use. Searching for new restaurants and services can be tedious and time consuming for yelp users. They must search numerous options and read multiple reviews per option to determine if it is a good fit. This process is a brute-force approach and often does not result in an optimal match. Here we use machine learning techniques to perform 3 experiments: first, we predict the star rating of a restaurant based on a set of reviews; second, we recommend a restaurant to a user based on their star ratings; third, we explore the potential of OpenAI in performing these tasks.

For all three experiments we utilize a dataset of restaurant reviews that has been made available by Yelp. For the first component we will provide a dataset of reviews and their corresponding star ratings. We feed the raw review text to a variety of classic machine learning algorithms (Naive Bayes, SVMs, multiclass logistic regression, KNN's, random forest classifiers), which output a numerical star rating from 1-5.

For the second component we input a set of user restaurant review ratings ("star ratings") from the Yelp dataset and use collaborative filtering techniques (nearest neighbor and matrix factorization) to identify similar users from which we make predictions (recommendations) of restaurants that users would likely enjoy visiting.

Finally, we explore OpenAI's text embeddings generated using the new and improved embedding model - *text-embedding-ada-002*. These embeddings will be used for the tasks such as 1) recommend top n restaurants relying on the similarity between embeddings of the user reviews, 2) semantic search by relying on the similarity between embeddings of the search query and user reviews text, and 3) multi-class classification tasks to predict the star ratings of the restaurant using embeddings of the user reviews.

2. Related Work

Current ML reccomendation systems fall into three categories: Collaborative Filtering (CF), Content Based (CB), and Hybrid Systems (HS)[1]. CF focuses on analyzing user behavior and preferences to identify patterns and recommend items based on similarities between users [1]. It relies on user-item interaction data, such as ratings or purchase history, and utilizes techniques like user-based or item-based similarity calculations to generate recommendations. In contrast, CB algorithms consider the attributes and characteristics of items to suggest similar items to users[2]. They analyze the content, such as text, tags, or features, associated with the items and match them to the user's preferences. CF is based on the assumption that people with similar tastes will rate items similarity, whereas CB is based on the assumption that items with similar features will be rated similarity[2].

CF techniques can be divided into two primary approaches: "neighborhood" methods and "modeling" methods[3]. CF neighborhood methods are based on computing similarities between people (users). For example, we would identify two similar users (Ui, Uj) and recommend items that Ui likes to Uj). Neighborhood methods include k-Nearest Neighbors (KNN) and k-Means (KM). Neighborhood techniques can have memory and computational challenges with large sparse datasets. CF modeling methods can be used to overcome the sparsity issues associated with CF neighborhood methods [4]. CF modeling methods use matrix factorization which also has the benefit of discovering and including "latent" (unknown) factors that are not represented in the dataset [5]. For example, a dataset on restaurants might not include information on the decor and/or cleanliness, yet these may be valuable "factors" in how people rate the restaurants. In theory, this ability to discover latent features can lead to improved performance compared to neighborhood techniques.

3. Dataset and Features

We utilize the publicly available Yelp dataset [6]. This dataset contains nearly 7 million reviews for over 150,000 businesses. Prior to conducting any analyses we performed various preprocessing techniques. We first converted our datasets into usable data formats, and filtered to select only restaurant data.. We merged information for duplicated restaurant records, and filtered out users and restaurants with no review history.

For our first component, (star prediction), we randomly selected 250,000 reviews from the set of all restaurant reviews. We did this to prioritize sample size over review and user history, as they are irrelevant for predicting star rating. These reviews were tokenized using CountVectorizer, and then scaled to avoid bias, prior to being used in any ML algorithm. We then split these reviews into a 200K train set and 50K test set.

For our second component,, we filtered the Yelp dataset to remove all users that < 10 reviews because for CF to perform well in identifying similar users it requires a meaningful number of reviews from a user. We test the performance on both large and small datasets. For the large dataset we filtered out restaurants with fewer than 100 reviews giving us a dataset of 1.5M reviews from 25,000 users of 7,000 restaurants. For the smaller dataset we limited it to include only restaurants in the Santa Barbara, CA area resulting in a dataset of 16,000 reviews from 1,000 users of 600 restaurants. Both datasets were randomly divided 80% for training and 20% for testing. It is important to note that the dataset was divided randomly by user to ensure that each user had 80% of their reviews in the training and 20% of their reviews in the test set. This is required to ensure the system is sufficiently verified as well as trained for each user.

For our OpenAI experiments we filtered the Yelp dataset to include only reviews for Santa Barbara CA. This is similar to the small dataset used in our second component except we did not filter to exclude reviews from users who had less than 10 reviews. Our resulting dataset included around 100,000 reviews of 7,024 restaurants which were divided 80%/20% between training and test sets for the classification task.

In our exploratory data analysis, we observed two things: First, that many users had fewer than 10 reviews and this has implications on the real-world applicability of user-based CF recommendation systems used alone. Second, most reviews were quite short. Due to this phenomenon, we chose to ignore stop words, as they would further shorten the length of reviews, making it difficult to conduct any analysis.

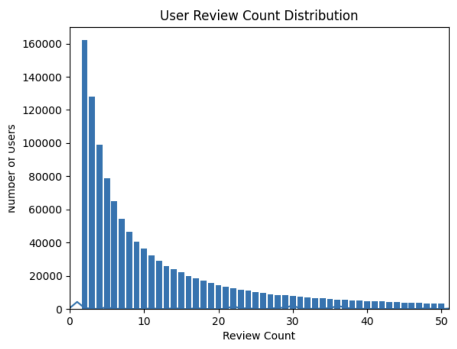

Figure 1 Distribution of number of reviews per user. Many users have left less than 10 reviews.

4. Methods

4.1 Star Prediction with Classical ML Classifiers

This section used a variety of classification algorithms: Naive Bayes, SVMs, multiclass logistic regression, KNN's, random forest classifiers (RFC)).

NB calculates the probability of a review having a certain rating based on the conditional probabilities of the words present in the review.

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

It assumes each word is independent of the rest.

SVM's attempt to draw classification boundaries around our reviews that maximize the margin between different classes (different ratings). They have the ability to transform non-linear data into a linear subspace, which is accomplished using kernels. For this project, we use linear SVM's to draw linear classification boundaries, as it is documented to handle large sparse datasets better than nonlinear SVM's [7].

Multiclass Logistic Regression (MLR) relies on reducing cross entropy loss, and uses the multinomial probability distribution to predict the chances of a review falling in a certain class.

Random Forest Classifiers use the bootstrapping, or bagging techniques to randomly select subsets of the training data to create numerous decision trees. These trees are averaged to create a merged, final prediction tree.

KNN algorithms are described further in section 4.2.

4.2 Collaborative Filtering (CF)

Our measure of error for our CF experiments is the commonly used root mean square error (RMSE) metric:

$$RMSE = \sqrt{\tfrac{1}{n}\sum_{u,r}(O_{u,r} - P_{u,r})^2}$$

Where O_{ur} is the observed rating of user u and restaurant r; and P_{ur} is the predicted rating of user u and restaurant r.

k-Nearest Neighbor

We use the k-nearest neighbor algorithm to identify similarity between users. A recommendation can be generated for a particular user by selecting items (in our case restaurants) that similar users have rated highly. In our experiments we utilize the cosine similarity measure:

$$\text{cosine\_similarity}(a, b) = \frac{\sum_i r_{ai} \cdot r_{bi}}{\sqrt{\sum_i r_{ai}^2}\sqrt{\sum_i r_{bi}^2}}$$

cosine_similarity(a, b) = (A · B) / (||A|| * ||B||)

We sum over the set of restaurants rated by users a and b (R_{ab}); r_{ai} is the rating of restaurant i by user a. A represents the vector or ratings of user a for the set of restaurants R_{ab}. ||A|| and ||B|| represent the L2 norm of vectors A and B respectively.

The hyperparameter k in k-NN represents the number of nearest neighbors to consider when making a prediction.

A lower value will result in the algorithm selecting fewer similar users and therefore give more weight to the ratings of the most similar users but may result in fewer recommendations and potentially more volatility. We shall test for the optimal values of k and use cross folding to avoid overfitting.

Matrix Factorization - SVD

Matrix Factorization takes the large (sparse) matrix of users x restaurants and decomposes into two lower-rank matrices of users x latent factors and restaurants x latent factors. The rating of a user for a restaurant (a cell in the large matrix) can be computed as the inner product of the two lower-rank matrices.

$m_{ij} = u_i \cdot r_j \quad \forall\ i, j$

$M = \{m_{ij}\}n_u \times n_r$ where M is the user x restaurant matrix and $n_u \times n_r$ are the number of users and restaurants.

$U = \{u_{ij}\}n_u \times n_d$ where U is the user x factor matrix and $n_u \times n_d$ are the number of users and factors.

$R = \{r_{ij}\}n_r \times n_d$ where R is the restaurant x factor matrix and $n_r \times n_d$ a the number of restaurants and factors.

RMSE is our error metric so the matrix factorization attempts to compute U and R by solving:

$(u_i , r_i) = \min \Sigma(O_{ur} - P_{ur})^2$

Computing the lower rank matrices can lead to overfitting the data [5] and to address this we will test two techniques to prevent the model from becoming too complex. First we will find a minimal value for the number of matrix factors (d), and second we will test adding a penalty to the model's loss function (regularization). Including the regularization penalty will modify the minimization problem to be: $(u_i , r_i) = \min \Sigma(O_{ur} - P_{ur})^2 + \lambda(\|u_i\|^2 + \|r_i\|^2)$, where λ determines the weight given to the regularization term.

We use Stochastic Gradient Descent (SGD) for our minimization algorithm. For matrix U, the update function is $u_i = u_i - \alpha(e_{ij}r_j - \lambda u_i)$. Where e_{ij} is the prediction error and α is the learning rate and λ is th regularization weight

4.3 OpenAI's text embeddings

OpenAI's text embeddings measure the relatedness of text strings. These embeddings are numerical representations of texts converted to number sequences, which makes it easier for ML models to understand the relationships between the texts. In other words, an embedding is a vector of floating-point numbers representing the respective text. The distance between two vectors measures the relatedness of the text. Small distances suggest high relatedness, and large distances suggest low relatedness. OpenAI's 2nd generation *text-embedding-ada-002* model promises better or comparable performance from all the old embedding models (1st generation davinci, curie, babbage models) and also has unified all five separate models (*text-similarity, text-search-query, text-search-doc, code-search-text, and code-search-code*) into a single new model.

Here we explore the use of *text-embedding-ada-002* in the context of recommendations, search, and classification tasks. We shall use OpenAI's text embeddings to find similarity between reviews and then predict and recommend restaurants for a given user.

5. Experiments / Results / Discussion

5.1 Star Prediction

We trained each of our algorithms on a set of 200,000 reviews, and tested their performance on a set of 50,000 reviews. We relied on precision to evaluate the models' success. Precision is defined as the proportion of correct predictions. This table reports the precision of each model in predicting each star rating, where the precision score represents the percentage of correct predictions in that class.

Star Rating	NB	SVM	MLR	KNN	RFC
1	0.47	0.53	0.7	0.54	0.71
2	0.19	0.28	0.46	0.26	0.48
3	0.23	0.29	0.44	0.25	0.44
4	0.38	0.40	0.48	0.29	0.39
5	0.70	0.68	0.7	0.5	0.57

Figure 2: Precision scores of each model's ability to correctly predict each star rating.

We note that all of our models struggle to differentiate between medium rated reviews, but achieve slightly higher performance for extreme reviews (1 and 5 stars). We believe this is due to the lack of nuance in these extreme reviews, while the medium rated reviews have several contradictions. We also believe that the length of the review may be influencing these scores. 5 star reviews tend to be long, as reviewers describe in detail what they enjoyed, which may be an influencing factor.

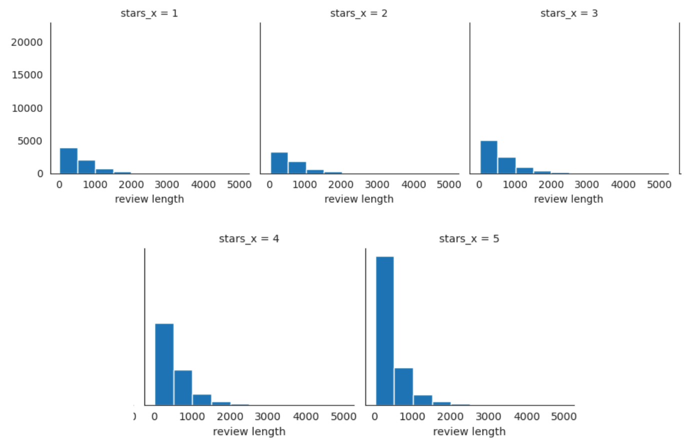

Figure 3: Distribution of the length of reviews. Most reviews are short, and 5-star reviews are the most common.

We conducted hyperparameter tuning (adjusting learning rates, changing k to larger number of neighbors, etc) but our efforts did not result in any significant changes in overall accuracy. However, our results do show that recommending highly rated restaurants may be an effective strategy, as most of our results perform decently well at deciphering 5 star reviews. We associate this trend to the fact that there are a higher number of 5 star reviews in our dataset.

5.2 Collaborative Filtering

Here we use CF techniques to identify user similarity and then make recommendations for a given user based on restaurants rated 5-stars by their set of similar users. RMSE measures the extent to which the recommended (predicted) restaurant matched the users actual (observed) rating for the restaurant. Figure 4 shows that the large dataset, k-Nearest Neighbor optimal performance was best for k=20 nearest neighbors[1]. For SVD Matrix Factorization the optimal number of latent factors (k) was found at 10. For MF, when k was increased the difference between training and test RMSE grew, likely indicating overfitting (varying the regularization constant λ did not significantly or positively affect the results).

k-Nearest Neighbor

Test	Dataset	Sample 1	Sample 2	Sample 3	Avg RMSE
	Train	1.072	1.033	1.016	1.04
kNN, k=5	Test	1.525	1.458	1.229	1.404
	Train	1.081	1.014	1.091	1.062
kNN, k=10	Test	1.372	1.375	1.15	1.299
	Train	1.009	0.924	1.007	0.98
kNN, k=20	Test	1.128	1.041	1.289	1.153

Matrix Factorization

Test	Dataset	Sample 1	Sample 2	Sample 3	Avg RMSE
MF, k=5, α=.01, λ=.02	Train	1.279	1.173	1.158	1.203
	Test	1.322	1.01	1.176	1.169
MF, k=10, α=.01, λ=.02	Train	1.216	1.077	1.172	1.155
	Test	1.12	1.018	1.174	1.104
MF, k=20, α=.01, λ=.02	Train	1.004	1.277	1.007	1.096
	Test	1.202	1.296	1.161	1.22

Figure 4:. CF performance on large dataset

Figure 4 shows that overall we were able to achieve our best solution using matrix factorization (1.104 on test). So the MF solution not only provided the benefit of scalability over the memory based kNN solution but also showed performance benefit, likely due to the discovery of "latent factors". The small difference between test and train RMSE indicates the solution was largely able to avoid overfitting.

In a real world scenario we would recommend 5-star rated restaurants, but it could be the case that for some users only 4 or 3-star recommendations are available. We investigated (see figure 5) and found that the kNN solution performed just as well when asked to return 5, 4, or 3 star recommendations (as measured by RMSE). In a practical application the system could be used to communicate that it thinks the users will "really like" (5-star), "like" (4-star) or "find acceptable" (3-star) a particular restaurant with equal confidence in each recommendation.

Dataset	5 stars	4 stars	3 stars
Train	0.980	0.9224	0.9661
Test	1.153	0.9931	1.055

Figure 5: Comparing performance across star ratings

For the smaller dataset consisting of 16,000 reviews from 1,000 users in the city of Santa Barbara, we were able to validate the full set of users and figure 6 shows that the k=10 kNN value provided the best results, while the MF solution performed best with k=20 latent factors. It is interesting to note that adding regularization or bias terms had slight negative effects, likely indicating any benefits of adding regularization to reduce overfitting are outweighed by the negative effects regularization can have on the model learning the underlying (latent) factors in the data.

k-Nearest Neighbor

Test	Dataset	RMSE
	Train	1.203
kNN, k=5	Test	1.169
	Train	1.155
kNN, k=10	Test	1.104
	Train	1.096
kNN, k=20	Test	1.220

Matrix Factorization

Test	Dataset	RMSE
MF, k=5, α=.01	Train	1.259
	Test	1.725
MF, k=10, α=.01	Train	1.219
	Test	1.203
MF, k=20, α=.01	Train	1.172
	Test	1.107
MF, k=40, α=.01	Train	1.142
	Test	1.203

Figure 6: CF performance on small Santa Barbara dataset

Our experiments show that the Yelp dataset lends itself well to CF user-based similarity recommendations. If a user has rated at least 10 restaurants we can recommend restaurants that the user will rate 4 or 5 stars and for small geographical areas the computations can be done quickly with kNN or MF solutions.

5.3 OpenAI text embeddings

OpenAI text embeddings are generated using the *text-embedding-ada-002* [10] model using OpenAI embeddings endpoint [8]. The use of OpenAI embeddings, experimentations, and approaches are deviced based on publicly available OpenAI Cookbook [9]. We used t-SNE to reduce the dimensionality of the embeddings to 2 dimensions. We also ran PCA for dimension reduction to 3 dimensions. The results are plotted for each review by its star ratings [1,2,3,4,5] in the figure 7. (For the t-SNE plot, 1=red, 2=darkorange, 3=gold, 4=turquoise, 5=darkgreen). Both plots suggest a mixed story of having a decent separation of star rating reviews on the extreme ends [1 and 5 ratings] yet there is a lot of overlap potentially making it difficult for recommendations, search and classifications tasks.

[1] We tested random samples of 100 users due to computational time constraints of the large dataset size.

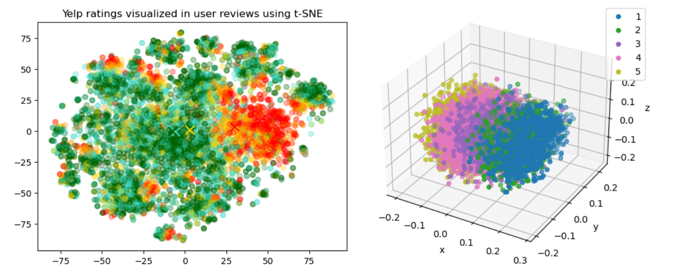

Figure 7: 2D & 3D representation of reviews using star ratings

5.3.1 Recommendation using OpenAI's text embeddings

We used our prior embeddings on our Santa Barbara dataset to recommend similar restaurants based on nearest-neighbor searches. First, we get the embeddings of the given user's reviews (each user is identified by UserId), calculate the distance between user's review and all other reviews, and then show the top n restaurants closest to the user's review. A sample result is shown below (key theme underlined) which has picked an authentic Mexican restaurant with 0.80 similarity to the user's review.

		Results		
UserId	Review (text)	Similarity	Restaurant	Review
W3CdwobBQu-xdfAIh4C07Q	This is the can't miss taco stop in Santa Barbara. Homemade corn tortillas served on the best steak, pork and chicken tacos you'll ever have the pleasure of devouring. Perfectly portioned, seasoned and priced. I've eaten a lot of tacos in my time here on Earth and these rank among the very best. Oh, and that salsa bar is still giving me daydreams.	0.8032655731019044	Taqueria Cuernavaca	I stopped here at my stop to Santa Barbara, coming down to LA from San Francisco. The food was amazing, and delicious. I would recommend anyone who like authentic Mexican food to stop and eat here. Good place.

Figure 8: Results of search using review of a specific user

5.3.2 Semantic Search using OpenAI's text embeddings

We also searched through reviews semantically by embedding the search query and then finding the most similar reviews. We compared the cosine similarity of the embeddings of the search query and all reviews and then selected top n best matches. Couple of sample results shown below, with the key theme of the query and result underlined. The semantic search has shown better results for both positive cases of identifying restaurants to visit and for negative cases about restaurants to avoid with around 0.86 and 0.83 similarity respectively.

Search Text	Results		
	Similarity	Restaurant	Review
I'm looking for a restaurant with a romantic ambiance and vegetarian meals.	0.8588487983833528	Thai Orchid	Land my husband went here for Dinner one day after work. The environment was nice and calm. Dimly lit and not a lot of noise at all. We are vegetarian and ordered Drunken noodles and Red curry along with spring rolls for appetizers. We loved the food. Asked them to make it spicy and it did come out pretty spicy! The service was great. Sicne its closer to work, I will definitely go back.
Show us top rated restaurants which are noisy to avoid them	0.832208489166308	Katie's Restaurant & Bar	The food is always excellent. However, it is very difficult to hold a normal conversation due to the abnormally loud interiors. Please please add some acoustic treatments to this space so we can enjoy the tasty food without a throbbing headache.

Figure 9: Results of semantic search

5.3.3 Classification using OpenAI's text embeddings

We also performed text classification using these embeddings, for this task we predict the star ratings of a restaurant using user reviews based on the embedding of the review's text. For proper performance evaluation on the unseen data, we did split the dataset into a training and test set. The plot below illustrates that the extreme reviews (5-stars and 1-star) show the best performance and are easier to predict as 2-4 stars reviews are expected to have more nuances.

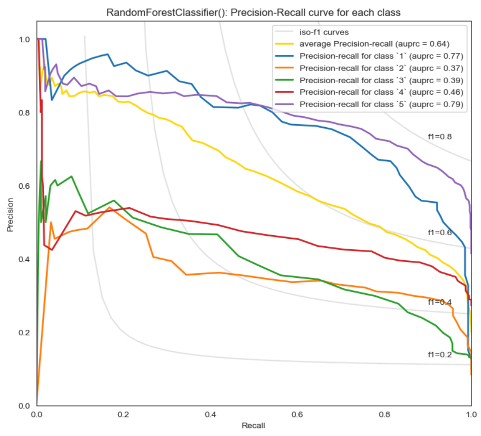

Figure 10: Precision-Recall curve for each star rating

Conclusion / Future Work

Our work has shown the feasibility of predicting restaurant "star" ratings from text reviews. We have shown that this works particularly well for highly rated (5 star) reviews which are typically more verbose. These highly rated restaurants are exactly what a recommendation system wants to recommend. Our experiments have shown that CF user similarity based recommendation based on user can perform well for large and small Yelp datasets. Future work could investigate combining these systems whereby ratings can be generated from text reviews and then these ratings could be used with user-based collaborative filtering techniques to make recommendations. Such a system can utilize CF (user similarity) methods when users have made sufficient reviews and CB (item similarity) methods when not (cold start problem). A particularly interesting area of investigation could be including other non-Yelp social media sources of restaurant reviews and experimenting with a small geographical area (a city for example) because the compute resources required are small and the matrix density is relatively high (many people having visited the same restaurants).

Contributions

Each team member contributed equally on a range of activities including project research, brainstorming, data analysis, experimentation, and report writing.

References

[1] Behera, Gopal, and Neeta Nain. "Collaborative Filtering with Temporal Features for Movie Recommendation System." Procedia Computer Science 218 (2023): 1366-1373.

[2] Schafer, J., Frankowski, D., Herlocker, J., & Sen, S. (n.d.). *Collaborative Filtering Recommender Systems*. https://faculty.chas.uni.edu/~schafer/publications/CF_AdaptiveWeb_2006.pdf

[3]Breese, J.S., D. Heckerman, and C. Kadie. *Empirical analysis of predictive algorithms for collaborative filtering.* In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI). 1998. Madison, Wisconsin. Morgan Kaufmann.

[4] Ajitsaria, Abhinav. *Build a Recommendation Engine With Collaborative Filtering* – Real Python. Realpython.com. http://realpython.com/build-recommendation-engine-collaborative-filtering/

[5] Koren, Yehuda, Robert Bell, and Chris Volinsky. *"Matrix factorization techniques for recommender systems."* Computer 42.8 (2009): 30-37.

[6]. Yelp Dataset. (n.d.). www.yelp.com. http://www.yelp.com/dataset

[7]. sklearn.svm.LinearSVC — scikit-learn 0.24.1 documentation. scikit-learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

[8] OpenAI API. platform.openai.com. https://platform.openai.com/docs/guides/embeddings/what-are-embeddings

[9]OpenAI Cookbook. GitHub. Published June 12, 2023. Accessed June 13, 2023. https://github.com/openai/openai-cookbook

[10] `Text-embedding-ada-002`: https://openai.com/blog/new-and-improved-embedding-model