Recipe search engine and recommendation function project update

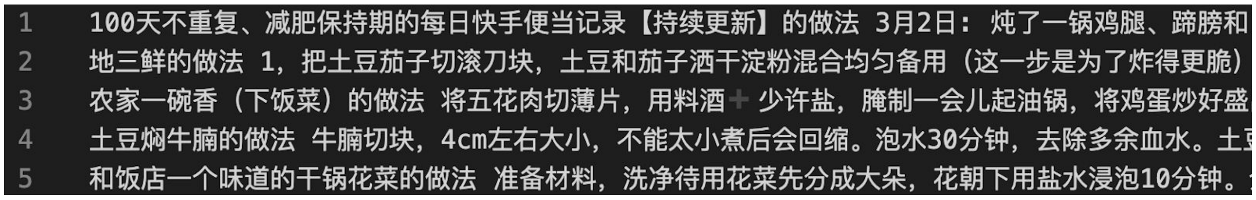
Zihan Tian

Introduction

The object of this project is to build a search engine for recipes searching and recommendation. Sometimes existing recipe websites always return recipes that are fancy or endorsed by many users because of the beautiful look of the dish. The instructions for thoses recipes may not clear or may be too fancy to conduct. And the suggestion system always recommends users recipes that are most popular but not meet the users' taste. It is meaningful to build a search and recommendation engine to retrieve and suggest the recipes that users are truly interested in. There are three main tasks in the first part of the project. First is to scrape recipes from a Chinese recipe website called xiachufang.com and make ground truth for the recipe record for 10 popular queries. The second task is to do word segmentation for Chinese recipes. The third task is to tune the information retrieval model to try to get more related and high scored recipe records.

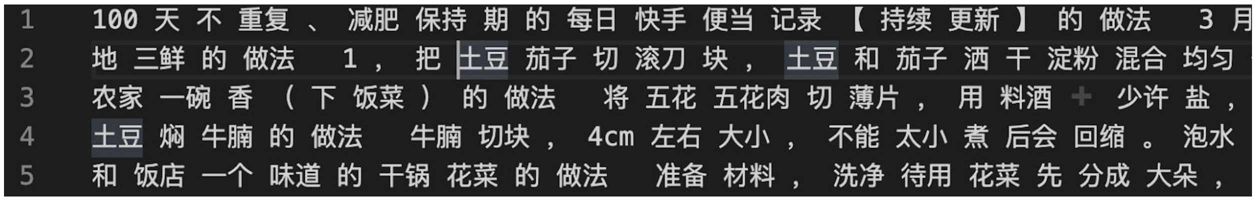
The second part is to construct a recommendation feature. When a certain recipe is selected, this recommendation feature should give 5 other recipes that the user might be potentially interested in.

Dataset



```
100天不重复、减肥保持期的每日快手便当记录【持续更新】的做法 3月2日：炖了一锅鸡腿、蹄膀和
地三鲜的做法 1，把土豆茄子切滚刀块，土豆和茄子洒干淀粉混合均匀备用（这一步是为了炸得更脆）
农家一碗香（下饭菜）的做法 将五花肉切薄片，用料酒 少许盐，腌制一会儿起油锅，将鸡蛋炒好盛
土豆焖牛腩的做法 牛腩切块，4cm左右大小，不能太小煮后会回缩。泡水30分钟，去除多余血水。土
和饭店一个味道的干锅花菜的做法 准备材料，洗净待用花菜先分成大朵，花朝下用盐水浸泡10分钟。
```

Picture 1: Sample data



```
100 天 不 重复 、 减肥 保持 期 的 每日 快手 便当 记录 【 持续 更新 】 的 做法  3 月
地 三鲜 的 做法  1 ， 把 土豆 茄子 切 滚刀 块 ， 土豆 和 茄子 洒 干 淀粉 混合 均匀
农家 一碗 香 （ 下 饭菜 ） 的 做法  将 五花 五花肉 切 薄片 ， 用 料酒  少许 盐 ，
土豆 焖 牛腩 的 做法  牛腩 切块 ， 4cm 左右 大小 ， 不能 太小 煮 后会 回缩 。 泡水
和 饭店 一个 味道 的 干锅 花菜 的 做法  准备 材料 ， 洗净 待用 花菜 先 分成 大朵 ，
```

Picture 2: Processed sample data

The recipe data for this project was scraped from a Chinese recipe website called xiachufang.com. Each line of record in this recipe dataset is a recipe as shown in picture 1. I used the beautifulsoup package to scrape the website and picked out the instructions for each recipe. Those instructions are written in Chinese. I used the jieba package to process sentences to words and concatenate these words by an empty space. The sample processed dataset is shown is picture 2. The processed dataset was passed to metapy and I used BM25 function to create my baseline. There are 958 recipes from 14 categories such as 'Home Style Dishes', 'Popular Dishes', 'Breakfast', 'Meat' etc. For each query, the searching engine will pick up ten recipes. At the next step, I will evaluate the results.

For the query dataset, I collect 10 common queries that usually happened in Chinese recipe queries. I find 50 corresponding recipes for each query and score them with the following policy.

I finished the annotation and saved the ground truth file for further evaluating the searching result. The metric is as shown here.
1 - A detailed instruction on how to cook the dish and contain all ingredients in the query and match the cooking method in the query;
0 - Only contain part of the ingredient or the cooking method is not the same as described in query;
-1 - Ingredients and cooking method are not matching with the query;

Methodology

1. How to get all words in a recipe instruction.
   It is easy to get separate words in English article. However in Chinese, words are not separated by empty space, which makes it hard to get all words in a Chinese article. For example, given a sentence like “牛肉和西红柿”(translated as ‘beef and tomato), how does a computer know to correctly separate the phase to “牛肉 和 西红柿” rather than others is the issue here. Here we use the jieba package to address this problem. Jieba searching engine mode separating function will take the input sentences or sentence and return a list of words. I processed this list and concatenated these words together and separated them by an empty space. I fed the reprocessed document to metapy, which makes it easy to get all terms and build the inverted index of the recipes dataset.
2. How to find and tune a proper information retrieval model.
   There are multiple models that could be used for scoring the documents. Here I used BM25 for baseline and I tried several ways to tune and build new functions. Below is the baseline BM25 function, where qtf is the term frequency in the query, tf is the term frequency in document, l_d is the length of the document, avg_l is the average length of documents, N is the number of documents, df is the document frequency and others are the parameters.

   $$BM25_{score}(Q,d) = \sum_{t\in Q} w(t,d)$$

   $$w(t,d) = \frac{qtf}{k_3 + qtf} \times \frac{k_1 \times tf}{tf + k_1(1 - b + b \times l_d/avg\_l)} \times log_2 \frac{N - df + 0.5}{df + 0.5}$$

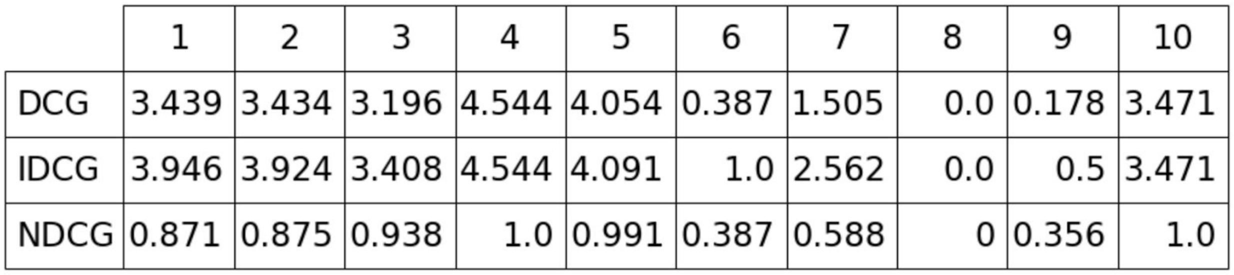
   One way to tune BM25 function is to consider changing parameter b. Parameter b has influence on the penalty of document length. Generally speaking, when b is set to 0.75, the searching engine will give a relative good performance. If b is set to 0, the length of the document will have no influence on it. Considering the aim of the recipe searching engine is to provide doable and concise instructions, which means the procedure could not be too complicated or too blur. I set up experiments for b changing from 0.50 to 0.8 and increase it 0.05 per time.The other way to get better results is to modify queries based on their characteristics and use different functions. For a concise recipe query, there may just contain the key ingredients in the query. In this case, I won’t need the first part in the original BM25 function to adjust the term frequency in a query. And for a long and detailed query, I chose to use the tuned BM25 function and remove the stop words from the query.
3. How to evaluate the retrieving results.

Normalized discounted cumulative gain at rank 10 is calculated for each query, results from other modified functions will be compared with the baseline BM25 results.

4. Calculate the similarity between documentations.
   A document can be represented by a vector in which each value is tf, the word frequency of a different term T in the document D. Keywords of a short document (such as news) are not many, so we can extract the keywords of each news, and use the TFIDF value of these keywords to constitute the vector representation of the document, which can greatly reduce the amount of similarity calculation, while maintaining a good recommendation effect. The jiaba package could return terms and corresponding TFIDF value. I collected the top 10 terms of each document to represent the document. I get the M*N matrix, where M represents the number of documents and N is the number of all selected unique terms in M documents. This matrix is a sparse matrix. I will use the pairwise distance method in sklearn and pandas package to calculate the cosine similarity and store the suggested 5 documents for each document as recommendation.

Evaluation and Results

1. Baseline query result.
   The method of evaluation is as described in the methodology part. Below is the baseline(BM25) results for 10 queries. (b = 0.5, k1 = 1.2, k3 = 500)



	1	2	3	4	5	6	7	8	9	10
DCG	3.439	3.434	3.196	4.544	4.054	0.387	1.505	0.0	0.178	3.471
IDCG	3.946	3.924	3.408	4.544	4.091	1.0	2.562	0.0	0.5	3.471
NDCG	0.871	0.875	0.938	1.0	0.991	0.387	0.588	0	0.356	1.0

Table 1: Evaluation results for 10 queries

2. Scores of experiments for parameter b.
3. Scores of experiments for modifying the query.
   It is hard to predict how the jieba package will do word segmentation at some time. For example, ‘麻婆豆腐’(Mapo tofu) which is the eighth query in the table above, may be separated as ‘麻婆 豆腐’ or treat it as a whole word and the jieba package won’t do word segmentation on it. So I tried to map the original query to a set of potential queries that could give better results. Those results are stored as extra recommendations if users don’t get the recipes they would like. Below is the table(Table 2) showing the results of using different query terms but actually query the similar recipe. The six queries are “麻婆 豆腐”, “豆瓣酱 豆腐”, “豆腐 肉末”, “豆瓣酱 豆腐 肉末” and “豆腐 猪肉”.

	1	2	3	4	5	6
DCG	0.0	0.25	0.0	0.158	0.538	0.0
IDCG	0.0	0.5	0.0	0.5	1.065	0.0
NDCG	0	0.5	0	0.315	0.505	0

Table 2: Evaluations of 6 queries for a same recipe

As shown in Table 2, the fifth query gave the best performance.

Work Plan

I have finished the data collection and processing procedure. I finished the code for building a basic searching engine, which is left to be tuned. I finished extracting the top 10 TFIDF terms for each document for the recommendation feature.

1. Finished the numerical experiment for tuning the BM25 functions.(1 week)
2. Finished calculating the cosine similarity of documentations. (2 weeks)
3. Report wrap-up. (2 weeks)

Related Work

As I stated above, I used the jieba package for Chinese word segmentation. There are several packages such as jieba, SnowNLP, THULAC, and NLPIR that can also do the similar job. The existing word segmentation methods can be divided into three categories: string matching based word segmentation method based on understanding and statistics based word segmentation method. The jiaba package used the third method because this method does not need very large quantity language knowledge and information and could do a great job on recognizing ambiguity.

The main statistical models are N-gram, Hidden Markov Model(HMM), ME and Conditional Random Fields(CFR) models. The jieba package utilizes the HMM and viterbi algorithm to calculate the result. And it turns out that the jieba package could output the result that I need for metapy.

There are mainly three types of recommendation engine. The first kind is a general recommendation engine. The administrator of a certain website will recommend the same things which is calculated to be a popular item among all users. The second kind of engine is based on the data source, such as demographic characteristics, tag of the items. The third kind of engine is based on the user-item two dimensions matrix that describe the user preference. Here in the project, I used a simple mechanism to calculate the suggested items. Because the dataset only contains the recipes but no users data there. In a complicated system, there might be needed to find similar users and similar items and utilize both of information. But here we just want to calculate similarity between documents. There are several ways to calculate the similarity such as Euclidean distance, Pearson’s correlation coefficient, Cosine similarity and Tanimoto coefficient . But all of them are vector based. And here I used Cosine similarity.

Reference
[1] Lin, Q.-X,Chang, C.-H. Chinese word segmentation using specialized HMM. In Proceedings of the 18th Conference on Computational Linguistics and Speech Processing, ROCLING 2006, pages 287-307, 2006
[2] Gaga Mao, Study on Chinese Word Segmentation, Advances in Higher Education, page 1, 2019
[3] Trotman, Andrew, Lilly, Kat, JASSjr: The Minimalistic BM25 Search Engine for Teaching and Learning Information Retrieval, pages 2185-2188, 2020
[4] Kalian, Kirk, Remig, Charles and Jung, Youna, BM25-AH: Enhanced BM25 Algorithm for Domain-Specific Search Engine, pages 631-634, 2019
[5] He, Ben, Ounis, Iadh, Term Frequency Normalisation Tuning for BM25 and DFR Models, In Lecture Notes in Computer Science, pages 200-214, 2005