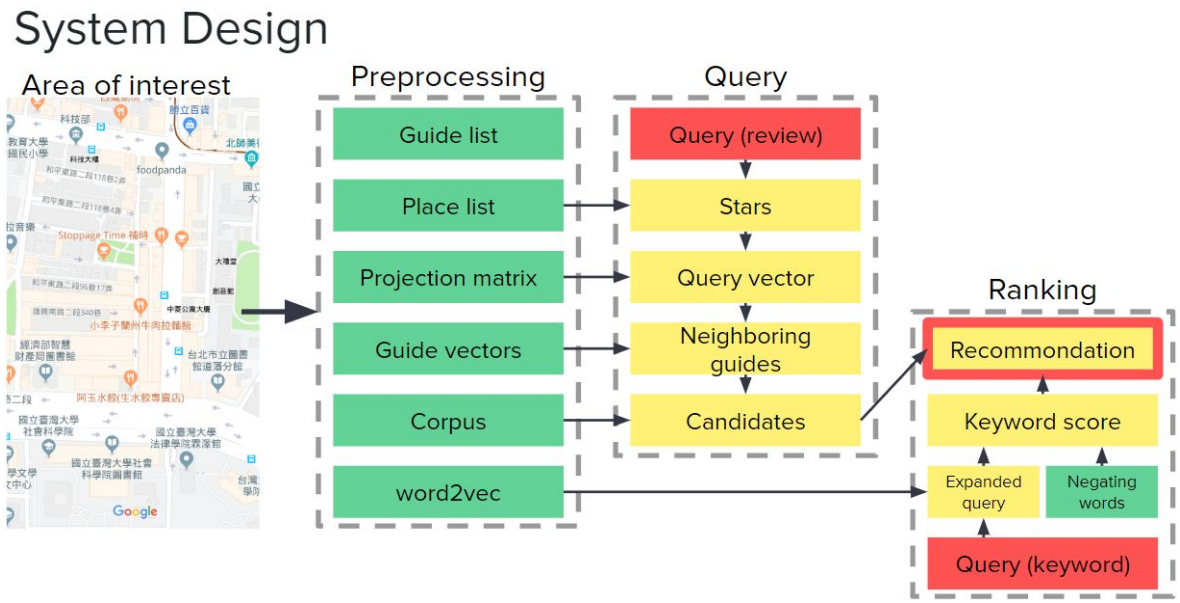
# DinnerSelector

Web Retrieval and Mining - Spring 2019 - Final Term Project

R07922096 莊昕寰 R07922101 游子緒 R07922137 王韻華 R07922139 蕭皓仁




## Motivation

Since it has always been a hard problem to decide what to eat for dinner in our lab, we want to solve it with IR techniques. Given a group of users, we aim to find the best recommendations with our system by analyzing their Google Maps review history.

## Data Collection

Descriptive or quantitative reviews to restaurants are publicly available on many platforms. For the areas near NTU, Google Maps has accumulated the most reviews so far. A review on Google Maps includes a number of stars (1-5, integer), an optional description, and a few optional pictures.

To find good references, we collect reviews from those who have lots of reviews around NTU. We call these reviewers guides. We crawled all the reviews from 92 guides and picked the places reviewed by more than 10 guides as our place set. The place set include non-restaurants places, such as **誠品書局台大店** and **國立臺灣大學綜合體育館**. We decided to keep these places because they are good indicators for differentiating reviewers.

## Preprocessing

The guides will first be converted into guide vectors according to the number of stars they give to the restaurant set. The values will be normalized so that each guide has the same mean (0) and variance (1). For unvisited restaurants, 0 will be assigned. By concatenating these values, a guide vector will be obtained, indexed by restaurants.

A matrix $X$ with its columns guide vectors will then be constructed. Since the 180-dimensional guide vectors are sparse, we project them into a 20-dimensional latent space by LSI. By taking the SVD of $X$, i.e. $X=U\Sigma V$, we can take the first 20 columns of $U$ as the projection matrix $P$. After projection, the guide vectors will be indexed by latent concepts, and restaurants with similar properties will have small Euclidean distance.

Description in the reviews of a restaurant will together be regarded as a document and segmented into words with *jieba*. All the documents will be regarded as the corpus, and the segmented words will be converted into word vectors with *word2vec*, which will be used in keyword expansion.

## Query

The query consists of a list of reviews from users and a few keywords. The stars in the reviews are converted into a vector in the latent space with the same process as the guides. After that, the 20 nearest guides are picked, and all the places reviewed by them are collected and counted. A score estimation is then conducted:

*score = 0.1 * review count*
*+ average(normalized #stars from guides)*

The top 30 non-restaurants places are picked as candidates and will be ranked with the keywords in the next stage.

## Ranking

We aim to rank the retrieved restaurant candidates with preferences of users. These preferences are given in the form of a list of keywords (e.g. 便宜).

First, keyword expansion is performed to score the candidates better. The trained *word2vec* model converts a keyword into its 100-dimensional vector. The 5 closest words (e.g. 實惠, 划算,...) in terms of Euclidian distance are added into the keyword list.

Next, all the review contents of a candidate are regarded as a document, and a corpus composed of all documents will be generated. The term frequencies of keywords are calculated for each document.

Some special cases need to be handled. If negating words (e.g. 不, 很差) are present in the context of the keyword, we will multiply the corresponding term frequency by *-2* to penalize.

The number of stars is a simple metric for ranking, but it's not accurate enough. After several trials, we use the following formula to estimate the likeliness a restaurant suits the users' taste.

*score = 0.1 * review count*
*+ 2 * average(normalized #stars from guides)*
*+ 100 * Σterm frequency / #keyword / #review*

## Experiments

Two sets of reviews are used to test the query stage. The first one is from the consensus of our lab (MVNLab), and the second one is from one of our team members (huzixiao).

Several words are used to test keyword expansion and the effect of negating words in context. Some of them are presented in the next section.

## Evaluation

For the query stage, MAP and precision of the 30 roughly ranked documents are evaluated. In the MVNLab query, the true positives and false positives are determined by votes from the four members; and in the huzixiao query, 蕭皓仁 marked the restaurants he is interested in. The results are summarized in the following table.

| query | MAP | precision |
|---|---|---|
| MVNLab | 43.2% | 36.7% |
| huzixiao | 46.1% | 56.7% |

For the effect of negating words, the accuracy of 3 words are evaluated. The confusion matrices are shown below.

"冷氣"

| Accuracy = 87.5% | Positive | Negative |
|---|---|---|
| Predicted positive | 55 (85.9%) | 7 (10.9%) |
| Predicted negative | 1 (1.6%) | 1 (1.6%) |

"乾淨"

| Accuracy = 92.1% | Positive | Negative |
|---|---|---|
| Predicted positive | 197 (86.4%) | 14 (6.1%) |
| Predicted negative | 4 (1.8%) | 13 (5.7%) |

"發票"

| Accuracy = 92.8% | Positive | Negative |
|---|---|---|
| Predicted positive | 10 (71.4%) | 1 (7.1%) |
| Predicted negative | 0 (0.0%) | 3 (21.4%) |

## Conclusion & Future Work

Based on latent concepts of restaurant preferences by reviewers, our system can recommend restaurants preferred by people with similar tastes. Furthermore, keywords can be used to rank the results.

Deep Learning techniques in Natural Language Processing (NLP) may be suitable for dealing with negating words and synonyms. We didn't have enough time to explore these techniques, but they are worth trying.

Finally, we hope those who get troubled in finding a place for dinner can benefit from this project.

## Division of Work

R07922096 莊昕寰
- Data collection
- Evaluation

R07922101 游子緒 (Leader)
- Crawler
- Preprocessing

R07922137 王韻華
- Parsing
- Query expansion

R07922139 蕭皓仁
- Negating words
- K-nearest neighbor

## Signature