# Task 6: Mining Popular Dishes and Restaurant Recommendation

Xiaoming Ji (xj9)

# Overview

In this task, we are going to predict whether a set of restaurants will pass the public health inspection tests given the corresponding Yelp text reviews along with some additional information such as the locations and cuisines offered in these restaurants. Making a prediction about an unobserved attribute using data mining techniques represents a wide range of important applications of data mining.

In this task, we are going to explore TFIDF, Topic Model and Document Embedding text representations, and Naive Bayes, Logistic Regression, SVM and Boosting classification models in order to find the best combinations to solve this problem. We also evaluated Flair, a deep learning NLP package to check whether we can leverage state-of-the-art NLP techniques to improve our results.

# Data Preparation

For this task, we have the dataset (**hygiene.dat**) that contains concatenated reviews for 13299 restaurants. A label file (**hygiene.dat.labels**) have the first 546 restaurants’ binary labels, where a 0 indicates that the restaurant has passed the latest public health inspection test, and 1 means that the restaurant has failed the test. An 3rd file (**hygiene.dat.additional**) provides additional information regarding a restaurant.

For the review dataset, we applied the preprocessing steps as,

* Tokenization: Split the sentences into words. Lowercase the words and remove punctuation. Words that have fewer than 2 characters are removed.
* All stopwords are removed.
* Words are lemmatized. Words in third person are changed to first person and verbs in past and future tenses are changed into present.
* Words are stemmed. Words are reduced to their root form.

To leverage extra info in **hygiene.dat.additional**, We applied the following steps,

* Cuisines: concatenate all cuisines as one string.
* Zipcode: convert to alphabetic characters (for example: 98009 -> “JIAAJ”) so that it won’t be removed by review preprocessing.
* Number of reviews: based on the distribution of number of reviews, we convert the number using formulas as,
  + (0, 2]: “NoReviews”
  + (2, 6]: “FewReviews”
  + (6, 13]: “SomeReviews”
  + (13, 50]: “ManyReviews”
  + [50, unlimited]: “LotReviews”
* Rating: based on the distribution ratings, we convert the number using formulas as,
  + (0, 2.8]: “PoorStars”
  + (2.8, 4.4]: “StandardStars”
  + (4.4, 5]: “GoodStars”

We then concatenate each field (for example: “vietnamese, sandwiches, restaurants, StandardStars, FewReviews, JIAAJ”) and added them to the review text. By doing this, we can use same text presentation and classification techniques to handle such extra info.

# Document Representation

Classification models won’t able to handle words, thus we need to convert each review document as a vector.

* Word Count: this is basic bag-of-words approach that represents a document by counting word frequency. *CountVectorizer* in scikit-learn python package is used for this task.
* TFIDF: a bag-of-words approach that uses TFIDF algorithm to calculate weight of a word. *TfidfVectorizer* in scikit-learn python package is used for this task. Besides the regular TFIDF, we also tried (2,3) word level n-gram and (2,3) characters level n-gram.
* Topic Model: we use LDA algorithm ()
* Doc2Vec:

# Classification Models

# Deep Learning Model