Cuisine Name Detection
(Analysis of Yelp dataset)

1. Data Preparation

For this task I analyze Indian cuisine.
The dataset contains 42,153 businesses, of which 202 are Indian restaurants (identified by presence of 'Indian' category).
After restricting businesses only to 'Indian' category, the dataset has 8,230 reviews for the Indian cuisine restaurants.
On import I converted each review to lower case, and removed punctuation.

2. Annotations File (Task 3.1 and baseline for Task 3.2)

I started with editing the annotations file for Indian cuisine, removing all non-dish lines. This left 23 dish names in the annotations file. I then converted the file to format required for Task 3.2 and submitted it, getting the score of 5.

3. ToPMine (improving results for Task 3.2)

I used ToPMine algorithm (based on *Scalable Topical Phrase Mining from Text Corpora* paper by Ahmed El-Kishky, Yanglei Song, Chi Wang, Clare R. Voss and Jiawei Hanto) to create a bag-of-phrases. The algorithm combines frequent phrase mining stage that collects statistics for candidate phrases, weeding out phrases that fail minimum frequency criteria with phrase construction phase that re-builds phrases from bottom up, using most significant candidates (phrases of shorter length). This is a very interesting approach, giving higher performance than regular n-gram approach, and potentially more meaningful results due to phrase construction from highest significance candidates rather than just frequency of the outcome. There is also topic modeling part of the algorithm, which I will not cover, since for this exercise I am only interested in bag-of-phrases creation.
Settings used to run ToPMine tool:

- minsup=10 (ignore phrases occurring less than 10 times)
- maxPattern=4 (maximum number of words in the phrase - dish names are at most three words, I included 4 just in case)
- gibbsSamplingIterations=0 (since I am using ToPMine only to generate bag-of-phrases, not for topic modeling)
- everything else has been left at default values

This resulted in a set of 1,588 phrases. Review of the phrases showed that some phrases were related more to quality than to dishes ("pretty good", "good food", etc.).
I constructed a special set of stopwords for words that should not appear in dish names: ['indian', 'india', 'food', 'lunch', 'dinner', 'we', 'like', 'love', 'good', 'great', 'be', 'is', 'was', 'to', 'are', 'recommend', 'recommended', 'ordered', 'have', 'had', 'back', 'staff', 'service'].
I excluded every phrase that contained any of these words, and merged the resulting list with the list from Annotations file (removing any duplicates), ending up with 838 phrases, which contained dish names with very few exceptions.
Submitting this file initially gave me a score of 8.
However, I have found a defect in the scoring algorithm. Some of the phrases produced by ToPMine contained two spaces between words, resulting in a number of duplicate dish names. When I removed double-space and re-submitted, the score dropped back to 5, which means that creators of the scoring algorithm likely missed this double-space problem.

4. SegPhrase (further improving results for Task 3.2)

I used SegPhrase tool (https://github.com/shangjingbo1226/SegPhrase, based on *Mining Quality Phrases from Massive Text Corpora* paper by Jialu Liu, Jingbo Shang, Chi Wang, Xiang Ren and Jiawei Han). The algorithm is also multi-phase: first it parses the corpus, computing phrase frequencies and generating candidate phrases; then it estimates phrase quality by calculating a number of features for the phrases, based on concordance (features quantifying probability that elements of the phrase occurred together as a phrase rather than as parts of different adjacent phrases) and informativeness (occurrence of stopwords within a phrase, average IDF of words in the phrase, and punctuation of the phrase – whether it's in quotes, brackets or capitalized), and uses a predictive model to calculate phrase quality; then it re-parses the corpus, assigning each frequent word occurrence to only one candidate phrase (segmentation), evaluating for segmentation that results in highest phrase quality, and re-calculating phrase frequencies according to segmentation (rectified frequencies); rectified frequencies result in new features that can be used in previously constructed predictive model to adjust phrase quality and perform new segmentation – this can be done for a number of iterations.
Settings used in running SegPhrase tool:

- AUTO_LABEL=1 (automatically generate labels)
- WORDNET_NOUN=1 (use nltk's wordnet corpus to keep only noun phrase candidates)
- SUPPORT_THRESHOLD=10 (ignore phrases occurring less than 10 times)
- MAX_ITERATION=5 (number of training iterations)
- Everything else has been left at default values

This resulted in a set of 1,513 phrases. Review of the phrases showed similar issues to those in ToPMine output. Additionally, SegPhrase included phrases with both high-quality scores and low-quality score in the output. Review has shown that quality score of 0.3 is a reasonable threshold.
I processed SeqPhrase output, removing phrases with scores below 0.3 and phrases that contained any stopwords from my special set, and merged the resulting list with the combined ToPMine/Annotations list (removing any duplicates), ending up with 1115 dishes.
Submitting this file raised the score from 5 to 9.

5. Sidekick pattern (further improving results for Task 3.2)

Next I decided to use an approach that is not strictly an algorithm, but more of a design pattern for solving problems with dirty data (based on Strata talk *The Sidekick Pattern: Using Small Data to Increase the Value of Big Data* by Abe Gong). The main point of the pattern is to find an external, small, well-curated dataset that can be used in conjunction with the dataset one is trying to analyze. Well, what better source of Indian cuisine names than Wikipedia?
I got the list of Indian dishes from the following Wikipedia pages:

- https://en.wikipedia.org/wiki/List_of_Indian_dishes
- https://en.wikipedia.org/wiki/List_of_snack_foods_from_the_Indian_subcontinent
- https://en.wikipedia.org/wiki/List_of_Indian_pickles
- https://en.wikipedia.org/wiki/List_of_Indian_breads
- https://en.wikipedia.org/wiki/List_of_Indian_beverages

This gave me a curated set of 521 dish names. However, I needed to be sure that these dish names actually occur in the reviews. Examining some of the reviews, I saw that many dish names in the reviews had extra words inserted in the middle of dish name, some 2- and 3-word dish names were mentioned in the reviews with just one word (and not always the same word), some words in dish names were misspelt in some of the reviews. So, just matching full dish names to reviews to find which dishes from curated data set are mentioned in the reviews clearly would not work.
Instead, I created a bag-of-words from the reviews, and considered each dish in the curated data set matched if every word in the name of that dish was present in the bag-of-words. This left 201 dish names.
I merged these 201 dishes with the combined SegPhrase/ToPMine/Annotations list (removing any duplicates), ending up with 1259 dishes.
Submitting this file gave me a score of 12.

6. Results Review

The most interesting thing for me is the difference between the curated dataset of 521 dish names and the mined dataset of 1259 dish names.
There are several reasons for this I identified:

- a lot of duplication comes from incomplete dish names in the reviews (“naan” and “naan bread”, “chicken tikka” and “chicken tikka masala” and “tikka masala”).
- dish variations contribute as well (“garlic naan”)
- many non-dish names that got through my filtering (“huge fan”, “main courses”, “las vegas”)

The first impulse may be to use only the intersection between curated dataset and mined dataset (201 dish names) and consider that to be the resulting list of dishes. However, that would be a mistake, because it would get rid of dish variations, and we cannot consider this to be a complete data set, there are many more dishes in Indian cuisine that are missing from the curated data set.
The second impulse is to try tuning ToPMine and SegPhrase to get rid of non-dish names. Both tools can be tuned, but we have to remember that neither one is designed specifically for dish-name mining. Both are designed for phrase mining and both will turn up phrases commonly used in the reviews.
So, we need a method to eliminate phrases commonly used in reviews that are not dish names. SegPhrase’s ability to use wordnet corpus to limit candidate phrases to noun-based ones is a big help, but we already used it and still SegPhrase’s output contains fare share of non-dish names.
Using stop-words also helps, but manually constructing a stop-word list is a tedious exercise.
One approach that may work is to use the rest of the reviews, and apply approach similar to TF-IDF: discard phrases that occur frequently in the reviews for Indian cuisine and are also frequent in reviews of other cuisines, keeping only phrases that occur frequently in Indian cuisine reviews.
I tried this using Italian cuisine reviews, and was able to reduce the Indian cuisine list to 693 dishes, a set with much higher quality than previous one, though still not perfect.
Submitting the result got a score of 10, which does not speak highly of the quality of the scoring function, which is consistent with my earlier finding when looking at scoring of ToPMine results.

7. Tools Used

Data processing: Python, ToPMine, SegPhrase

8. Important code snippets

Filtering Indian cuisine:

```
restaurant_ids = []
with open(path2buisness, 'r') as f:
    for line in f.readlines():
        business_json = json.loads(line)
        if 'Indian' in business_json['categories']:
            restaurant_ids.append(business_json['business_id'])
```

Importing reviews, and creating bag-of-words for later curated dataset merge:

```
reviews_text = []
words = set()
with open (path2reviews, 'r') as f:
    for line in f.readlines():
        review_json = json.loads(line)
        if review_json['business_id'] in restaurant_ids:
            text = re.sub(r"\W+", " ",
                          review_json['text'].lower().encode('ascii', 'ignore').translate(None, string.punctuation))
            for word in text.split(' '):
                if word != '':
                    words.add(word)
            reviews_text.append(review_json['text'])
```

Import of labels file:

```
dishes = []
with open('task3/Indian.label', 'r') as f:
    for line in f.readlines():
        if line not in dishes:
            dishes.append(line.split('\t')[0])
```

Import of ToPMine output, filtering by stopwords, space compression and merge with labels dataset:

```
exclude = set(['indian', 'india', 'food', 'lunch', 'dinner', 'we', 'like', 'love', 'good', 'great', 'be', 'is', 'was',
               'to', 'are', 'recommend', 'recommended', 'ordered', 'have', 'had', 'back', 'staff', 'service'])
with open('task3/topPhrases.txt', 'r') as f:
    for line in f.readlines():
        line = line.split('\t')[0]
        line = ' '.join([x for x in line.split(' ') if x != ''])
        wordset = set(line.split(' '))
        if len(wordset) <= 3 and len(exclude.intersection(wordset)) == 0 and line not in dishes:
            dishes.append(line)
```

Import of SegPhrase output, filtering by score and stopwords, space compression and merge with ToPMine/labels dataset:

```
with open('task3/salient.csv') as f:
    for line in f.readlines():
        line,score = line.split(',')
        if float(score) > 0.3:
            line = ' '.join([x for x in line.split('_') if x != ''])
            wordset = set(line.split(' '))
            if len(wordset) <= 3 and len(exclude.intersection(wordset)) == 0 and line not in dishes:
                dishes.append(line)
```

Filtering curated dataset to only dishes mentioned in reviews and merge with SegPhrase/ToPMine/labels dataset:

```
for dish in curated_dishes:
    if len(set(dish.split(' ')).intersection(words)) == len(dish.split(' ')):
        if dish not in dishes:
            dishes.append(dish)
```

9. Resulting data set (excerpt)

chicken tikka	tandoori chicken	rogan josh	gulab jamun
basmati rice	chicken tikka masala	white rice	naan
tikka masala	coconut chicken	garlic naan	mango lassi
butter chicken	chicken curry	naan bread	palak paneer
saag paneer	lamb vindaloo	goat curry	malai kofta
lamb curry	rice and naan	chicken korma	chicken vindaloo
mt everest	tandoori times	chana masala	curry chicken
vegetable samosas	chai tea	chicken tandoori	chicken biryani
taj mahal	chicken masala	aloo gobi	royal taj
bombay spice	chicken makhani	lamb korma	masala dosa
vegetable korma	delhi palace	garlic nan	curry sauce
naan and rice	vegetable pakora	veggie samosas	paneer tikka masala
chicken tikki masala	guru palace	taj palace	plain naan
nan bread	belly dancer	mosque kitchen	samosa factory
mint chutney	time i visited	medium spicy	veggie dishes
curry house	buffet items	garlic naan bread	chicken and lamb
mount everest	tamarind sauce	fish curry	vegetable curry
tandori chicken	chili chicken	fresh naan	flavor and spice
mixed vegetable	mango ice cream	curry corner	spices and flavors
copper kettle	mango custard	gobi manchurian	cheese naan