Task 3

Overview

The goal of this task is to mine the data set to discover the common/popular dishes of a particular cuisine. Typically, when you go to try a new cuisine, you don't know beforehand the types of dishes that are available for that cuisine. For this task, we would like to identify the dishes that are available for a cuisine by building a dish recognizer.

Task 3.1: Manual Tagging

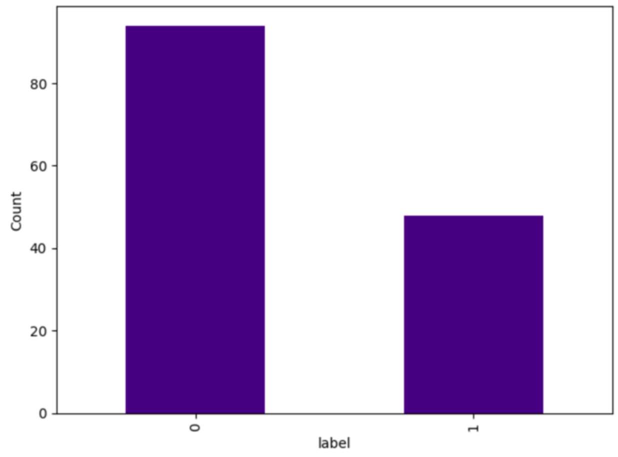
- Data set – manualAnnotationTask.zip
- Objective - Refine the label list for one cuisine
- Cuisine selected – Indian.label (character between a phrase and its label is **a tab instead of a space**.)

Actions taken

- Remove a false positive non-dish name phrase (recommended), e.g., hong kong 1 could be removed in Chinese cuisine.
- Change a false positive non-dish name phrase to a negative label, e.g., hong kong 1 could be modified as hong kong 0.
- Remove a false negative dish name phrase, e.g., wonton strips 0 could be removed in Chinese cuisine.
- Change a false negative dish name phrase to a positive label (recommended), e.g., wonton strips 0 could be modified as wonton strips 1.
- Add some new annotated phrases in the same format.

Procedure

- Shape of the data set: (142, 2)
- Labels count 0 – 94; 1 - 48
- Using 'label2csv.py' to convert to csv
    - Have used this file to scan through to correct/improve dish names and respective labels.
    - Applied 'lower case'
    - Output: 'indian_label.csv'



Observations

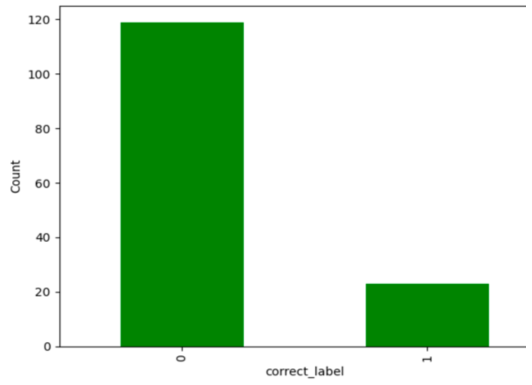
- Its' hard to differentiate these days that a dish is of any specific place/cuisine. There is regional version of pretty much anything in Indian cuisines.
- There are attributes like 'South India', 'Asian' which I have identified as non-dish names.
- Also, I did not apply 'know-it-all' approach. Have given lenient tolerance to categorize like 'tomato sauce', 'ice-cream', 'chicken wings', etc.
- Overall, I did not observe a great tagged/identified dish names in the given list.
- In 'False Negative' category, I am surprised, had only 2 dish names which I could recognize as valid

dishes 'tikka masala' and 'coconut chicken'.

- Corrected column name is saved as 'correct_label' and you can find the 'ground_truth' column too.

After manual annotation

- Shape of the data set: (142, 4)
- Labels count 0 – 119; 1 – 23



- There is drastic reduction in positive labels due to rectification.

Improvisation to list

As this can also be treated as part of "task 3.2" here. Nonetheless I improvised some knowledge base reference just in case I would need to use in automated dish extractions.

Source https://www.eatthis.com/popular-indian-dishes/

- Added few new annotated phrases in the same format to file 'indian_label.csv'
- 15 dishes added to the bottom of the file

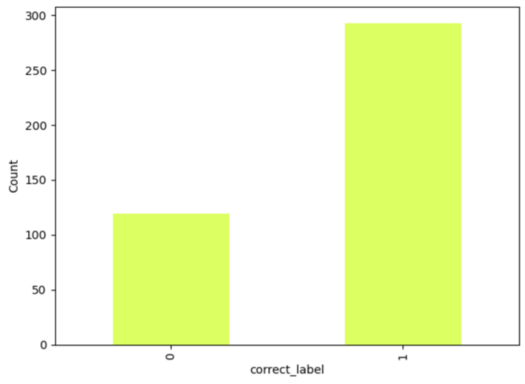
Source https://en.wikipedia.org/wiki/List_of_Indian_dishes

- 167 new dishes added to the list in an expected format

After improvisation

Shape of the data set: (412, 5)

- Labels count 0 – 119; 1 – 293



Task 3.2: Mining Additional Dish Names

I am planning to use suggested tools to continue my analysis and observations to complete this task.

I wanted to use the topMine, but when tried there were lots of files written in python which are outdated. I fixed few, but it required more work. Therefore, I went with text-mining based off of Java.

Textmining.jar

textmining.jar is a text mining tool with topic modeling and mutual information capabilities.

The topic modeling algorithm implemented is called "Comparative Text Mining". More information can be found at *"A cross-collection mixture model for comparative text mining, ChengXiang Zhai et. al., 2004"*

The Comparative Text Mining outputs common themes and collection-specific themes.

Experiment 1

- Dataset: Indian comments
  - This data was provided by course (placed in same dir)
- Number of Docs:19838
- Number of Unique Terms: 15112
- Clusters: 4
- Iterations:10

Cmd: java -cp textmining.jar topicmodels.CTMMultiRun param

I am pretty amazed how well the 4 clusters have been formed. The first 3 are not much relevant for dish name mining. However, 4th cluster is very useful.

Let's look at the 4th cluster from generated output file 'topics_termprobs'. Below I found it interesting and powerful output in terms of identifying the dishes, pretty fascinating!

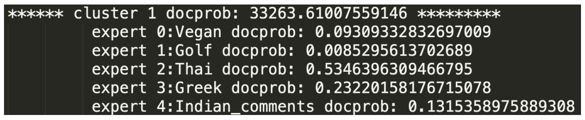
- chicken 0.07909712183997746
- rice 0.03146325698395835
- lamb 0.02773829807693564
- naan 0.02747333826541
- garlic 0.024863907026732496
- masala 0.02337023429488036
- sauce 0.022735388127238007
- curry 0.02210329573801954
- dish 0.016028887150561853
- ordered 0.013691486463973305
- paneer 0.011420029281907506
- flavor 0.00908280158468419
- bread 0.009062957622977137
- korma 0.008772278964612662
- tandoori 0.008537846745917036
- mango 0.008348361534028398
- vindaloo 0.008184442147215858
- fried 0.007148721671076005
- spicy 0.006574610022661328
- cooked 0.006551613439906799

Experiment 2

I want to understand what happens if we introduce another set of files from different cuisines. What will the output be like? Let's examine.

In the original zipped directory, there were 4 files containing the comments from different categories such as Golf, Greek, Thai and Vegan. So, I just executed this program with the same parameters.

Output was way more impressive. Each cluster has sub category as common model and "expert" model like categories for each of Indian, Golf, Greek, Thai and Vegan.



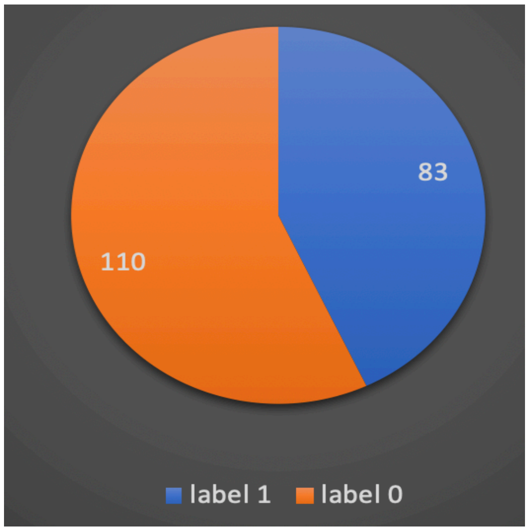
```
****** cluster 1 docprob: 33263.61007559146 *********
         expert 0:Vegan docprob: 0.09309332832697009
         expert 1:Golf docprob: 0.0085295613702689
         expert 2:Thai docprob: 0.5346396309466795
         expert 3:Greek docprob: 0.23220158176715078
         expert 4:Indian_comments docprob: 0.1315358975889308
```

- Cluster 1- with document probability 0.13
- Cluster 2 - with document probability 0.12
- Cluster 3 – with document probability 0.13
- Cluster 4 - with document probability 0.12
- Overall 51*4 clusters

Analysis

- Output-> 'textmining_java_label.csv'

- I cleaned the data by removing the duplicates and then analyzed if I could find dish names.
- Scanned through 197 unique items and found x items correct dish and then I labelled y items as ‘0’
- Also note that I found vegetable/meat names but since its universal, I let it be.
- Plus, there dishes which are spelled differently at different places, so I let them be. And also, words are not “stemmed”
- Observed more meaningful dish name sin cluster 4 than other clusters.
- If time permitted, I would have tried to see if it was possible to generate multiple word phrase.
- *Interesting find:* naan,lamb,paneer,tandoori,samosas,vindaloo,korma,dosa,saag,chutney,samosa,butter,biryani,masala,goat,aloo,basmati,vegetable,chutneys,pakora,dal,pudding,makhani,thali,sauces,gobi,palak,chai,kheer,peas,tamarind,plain,raita,sambar,pakoras,lentils,cauliflower,bread,chaat,manchurian,spinach,chana,tikki,daal,okra,josh,dosai,rogan,paratha,roti,naans



Word2Vec

https://radimrehurek.com/gensim/models/word2vec.html

https://rare-technologies.com/word2vec-tutorial/

```
ngramNum = 2
sentences = Sentences(inputPath)
ngram = None
for i in range(ngramNum - 1):
    ngram = gensim.models.Phrases(sentences)

model = gensim.models.Word2Vec(ngram[sentences], workers=6)
model.save(modelPath)
```

The link above explains well on this tool, so I am avoiding the repetitive points.

- I have used below code to generate (used nltk to process sentences) the model.
- Collected 175983 words types from a corpus of 377412 words (unigram + bigrams) and 42143 sentences
- training model with 6 workers on 5460 vocabulary and 100 features, with 5 epochs.

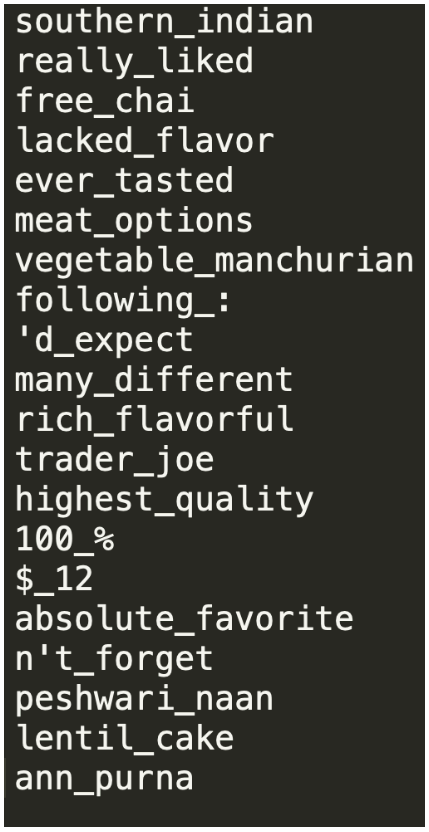
Analysis

```
model = gensim.models.Word2Vec.load(modelPath)
labelpos, labelneg = proceeseLabel(labelPath)
labelpos = set(labelpos) & set(model.wv.vocab.keys())
labelneg = set(labelneg) & set(model.wv.vocab.keys())

mostSim = model.most_similar(
    positive=labelpos,
    negative=labelneg,
    topn=100000
)

newDishes = [i[0] for i in mostSim if len(i[0].split('_')) > 1]
for dish in list(newDishes)[:20]:
    print(dish)
```

Let’s look at the generated phrases (I am ignoring their score, just analyzing the phrases).

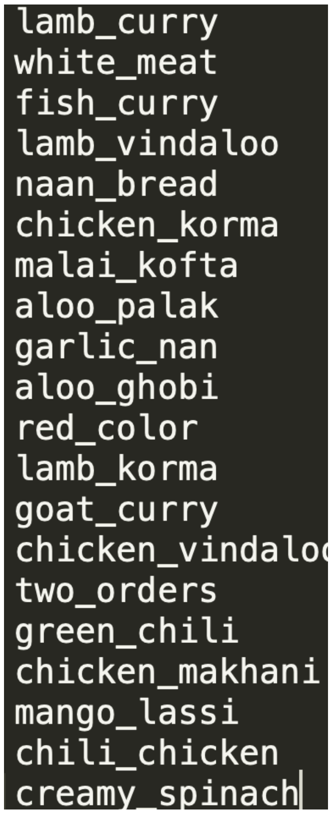


```
southern_indian
really_liked
free_chai
lacked_flavor
ever_tasted
meat_options
vegetable_manchurian
following_:
'd_expect
many_different
rich_flavorful
trader_joe
highest_quality
100_%
$_12
absolute_favorite
n't_forget
peshwari_naan
lentil_cake
ann_purna
```

With the default parameter set to the tool, it did not produce much valuable output. There are however phrases which make meaningful dish names such as 'vegetable_manchurian', 'peshwari_naan', 'lentil_cake', etc.

For above I had used the Indian.label from task 3.1. I wanted to test and see the output from the one that I manually enhanced. And below are the results.

The results are much better and closer to identifying Indian dishes. There are however few non-dish items here like 'red_color' and 'two_orders' Overall the result looks promising. It would be interesting to tweak other parameters to the function on lines of TFIDF, iterations, epochs, stemming, stop words and learn the quality of generations.



```
lamb_curry
white_meat
fish_curry
lamb_vindaloo
naan_bread
chicken_korma
malai_kofta
aloo_palak
garlic_nan
aloo_ghobi
red_color
lamb_korma
goat_curry
chicken_vindalo
two_orders
green_chili
chicken_makhani
mango_lassi
chili_chicken
creamy_spinach
```

Conclusion/opinion on generated results

If building a better tool to generate phrases, I would combine the power of multiple different tools, to make sure the edge cases are covered and a high-quality corpus or knowledge base is created.

But it's a potential area for restaurants to get the knowledge of what are the dishes people are enjoying and how they can benefit adopting to attract better strategies based on this information.