**Task 3: Dish Recognition**

**Jonathan Chang (jc26)**

**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**

In this task, I chose to focus on the Indian cuisine because it had the fewest number of candidate dish names, and I am not a huge fan of manual effort 😀. Per the instructions, I did a quick pass through the ~150 dish names, and created a column that contained my manual label which I considered "truth". I generally would label any phrase that was not actually a dish name as 0, and even if the phrase could be classified as "Indian," I still labeled them as 0 because as the overview states, we are particularly focused on identifying the dishes that are available for a particular cuisine. However, if a dish could be considered Indian cuisine at all (e.g. coconut chicken or tomato soup), I would assign a 1.

Per the suggestions, I removed all the false positives that I identified (30), and I corrected the labels of all the false negatives (5).

The result of this task was a file called ***Indian\_updated.label*** which I placed in the /data directory for SegPhrase section

**Task 3.2 Mining Additional Names**

**Discussion:**

I chose to use SegPhrase as opposed to ToPMine or word2vec primarily because it is recommended by the instructions as state-of-the-art. I also liked the idea of my manual effort from Task 3.1 being put to good use. As mentioned in the instructions, "SegPhrase has a classifier to assign a quality score to each phrase candidate based on their statistical features. The classification procedure will be enhanced by phrasal segmentation results. These two parts could mutually enhance each other."

As I understand it, ToPMine works well when we are trying to perform phrase mining without training data, however sometimes a small set of training data may enhance the quality of our phrase mining, and we should be able to improve our results by utilizing SegPhrase with our labeled phrases in conjunction with a constructed Knowledgebase.

SegPhrase is designed for phrase mining with tiny training datasets such as this problem, and it is able to use labels (such as our human annotated labels or from a knowledgebase) to indicate whether a phrase is high quality or not. It then performs classification via the random forest algorithm to construct models that distinguish between quality phrases and poor ones. It also uses phrasal segmentation to determine which phrases are more appropriate and partitions a sequence of words by maximizing likelihood. Overall, SegPhrase is fits the bill for pattern mining for this task because 1) we have a small training set 2) we need some way to perform feature extraction and distinguish how informative or quality a phrase is, and SegPhrase can utilize IDF and mutual information in order to mine relevant dishes and filter out irrelevant dishes.

**Setup:**

To set up SegPhrase on my Mac, I first cloned the repo from: <https://github.com/shangjingbo1226/SegPhrase>. I also needed to do some additional work to be able to build SegPhrase with the provided Makefile which included, per a helpful Github Issues Post referenced below:

1. Installing Xcode on my Mac (<https://stackoverflow.com/questions/19580758/gcc-fatal-error-stdio-h-no-such-file-or-directory>)
2. using brew to install gcc49
3. Changing the g++ variable in my Makefile to reference my gcc path (export CXX = /usr/local/Cellar/gcc@4.9/4.9.4\_1/bin/g++-4.9)
4. created a conda environment for python 2.7 and installed scikt-learn and nltk (which i need to activate when running the shell scripts)

Next, I used Wikipedia to generate a Knowledgebase of common Indian dishes

<https://en.wikipedia.org/wiki/List_of_Indian_dishes#Unsorted>. I added 250 Indian dish names I found from that Wikipedia article and appended it to the bottom of /data/EN/wiki\_quality.txt (from AutoPhrase repo) to create /data/wiki\_labels\_quality\_append\_IndianDish.txt.

I then proceeded to modify the provided train\_toy.sh script, by changing some of the following parameters. For RAW\_TEXT, I used the Indian.txt from Task 2 where we have all the Yelp reviews related to Indian cuisine. For KNOWLEDGE\_BASE, I used the knowledgebase created from the Wikipedia article as described above.

**RAW\_TEXT='data/Indian.txt'**

**AUTO\_LABEL=0**

**WORDNET\_NOUN=0**

**DATA\_LABEL='data/Indian\_updated2.label'**

**KNOWLEDGE\_BASE='data/wiki\_labels\_quality\_append\_IndianDish.txt'**

**KNOWLEDGE\_BASE\_LARGE='data/wiki\_labels\_all.txt'**

Then I could just run ./train\_IndianDish.sh

**Take 1**

First, I tried using the following data labels as described above. We see that although SegPhrase does a pretty good job with output phrases, it still has quite a few false positives and struggles with some of the ambiguous categories where the phrase is related to Indian culture, but not specifically cuisine.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Take 1 | |  | Take 2 | |
| fried\_rice | 0.998857168 |  | rice\_pudding | 0.999156153 |
| tandoori\_chicken | 0.99792519 |  | chicken\_tikka\_masala | 0.999156153 |
| rice\_pudding | 0.997713903 |  | butter\_chicken | 0.999156153 |
| indian\_cuisine | 0.997379549 |  | indian\_cuisine | 0.997890711 |
| goat\_curry | 0.997351527 |  | tandoori\_chicken | 0.99674297 |
| basmati\_rice | 0.997261925 |  | palak\_paneer | 0.996644708 |
| hot\_sauce | 0.99715519 |  | basmati\_rice | 0.995076304 |
| rogan\_josh | 0.996804812 |  | tikka\_masala | 0.993038088 |
| gulab\_jamun | 0.996804812 |  | chicken\_tikka | 0.993038088 |
| flat\_bread | 0.996391304 |  | fried\_rice | 0.989795259 |
| lamb\_vindaloo | 0.996236284 |  | flat\_bread | 0.986854083 |
| the\_naan\_was | 0.995780933 |  | the\_naan\_was | 0.984043204 |
| ice\_cream | 0.995741925 |  | gulab\_jamun | 0.982393454 |
| chicken\_tikka | 0.995741925 |  | ice\_cream | 0.979565057 |
| bhindi\_masala | 0.995527236 |  | the\_indian\_sampler | 0.979370539 |
| tikka\_masala | 0.995261925 |  | coconut\_chicken | 0.97706684 |
| south\_india | 0.995075258 |  | hot\_sauce | 0.97684241 |
| mother\_india | 0.994941925 |  | south\_india | 0.976455586 |
| tomato\_soup | 0.994107588 |  | mother\_india | 0.976359002 |
| garlic\_naan | 0.993636662 |  | garlic\_naan | 0.973038088 |
| chicken\_tikka\_masala | 0.991797101 |  | rogan\_josh | 0.969384489 |
| chick\_peas | 0.99135218 |  | tomato\_soup | 0.967324741 |
| coconut\_chicken | 0.990149853 |  | lamb\_vindaloo | 0.966972855 |
| veggie\_korma | 0.98840519 |  | goat\_curry | 0.966817936 |
| curry\_house | 0.98840519 |  | spice\_level | 0.962506452 |
| south\_indian | 0.986716284 |  | chick\_peas | 0.96123202 |
| masala\_dosai | 0.985939327 |  | lunch\_buffet | 0.95382734 |
| lunch\_buffet | 0.985713903 |  | paneer\_tikka\_masala | 0.939596421 |
| date\_night | 0.985276062 |  | india\_oven | 0.895114952 |
| curry\_houses | 0.985276062 |  | eggplant\_dish | 0.895114952 |
| mango\_chutney | 0.97840519 |  | chana\_masala | 0.894532005 |
| india\_oven | 0.975728039 |  | weekend\_buffet | 0.894458144 |
| mantra\_masala | 0.967741925 |  | chilli\_chicken | 0.894458144 |
| saag\_paneer | 0.967261925 |  | small\_portions | 0.876854083 |
| shrimp\_vindaloo | 0.965939327 |  | lunch\_hour | 0.868971602 |
| channa\_masala | 0.965276062 |  | bhindi\_masala | 0.866854083 |
| the\_indian\_sampler | 0.963941898 |  | menu\_items | 0.850478774 |
| yogurt\_sauce | 0.95840519 |  | saag\_paneer | 0.845609219 |
| star\_review | 0.95840519 |  | mango\_ice\_cream | 0.84359657 |
| brown\_rice | 0.95840519 |  | south\_indian\_dishes | 0.843173786 |
| eggplant\_dish | 0.956391304 |  | naan\_bread | 0.838366902 |
| chicken\_wings | 0.956299927 |  | buffet\_style | 0.832993652 |
| tandoori\_times | 0.956129549 |  | india\_masala | 0.829482784 |
| customer\_service | 0.956129549 |  | medium\_spice | 0.825114952 |
| white\_meat | 0.955939327 |  | chicken\_tikki\_masala | 0.824031637 |
| chicken\_tikki\_masala | 0.954611021 |  | chicken\_tika\_masala | 0.817067351 |
| masala\_dosa | 0.947831527 |  | chicken\_pakora | 0.816362192 |
| mutter\_paneer | 0.947741925 |  | lentil\_soup | 0.809795259 |
| mattar\_paneer | 0.947741925 |  | masala\_dosa | 0.782792875 |
| coconut\_chutney | 0.947741925 |  | spice\_levels | 0.781108777 |
|  |  |  |  |  |
| **12 false positives** |  |  | **13 false positives** |  |

**Take 2**

I wanted to see if I could improve my results so I created a new file called: Indian\_updated2.label. Here I decided to make two changes:

* I corrected the false positive dish names instead of removing them (I noticed that the false positives made their way back in)
* I added some new positive labels based on a Google Search of "indian dishes." (I took the top 25)

1. biryani
2. chicken tikka masala
3. samosa
4. butter chicken
5. tandoori chicken
6. panipuri
7. naan
8. tulab jamun
9. chapati
10. dosa
11. laddu
12. momo
13. rasgulla
14. papadum
15. chana massala
16. korma
17. palak paneer
18. khichdi
19. dkhola
20. pakora
21. idli
22. appam
23. pav bhaji
24. raita
25. kheer

I changed **DATA\_LABEL='data/Indian\_updated2.label'** and ran ./train\_IndianDish.sh again. Now, we take a look at the top 50, and there is no noticeable improvement, although we do mine a few other dishes that may not have been discovered in our first run. I suppose that I could continue to iterate and improve my results by manually correcting the output results for the false positives which is what makes SegPhrase awesome because it can be an iterative process with a feedback loop for improvement.

**References**

* Jialu Liu\*, Jingbo Shang\*, Chi Wang, Xiang Ren and Jiawei Han, "[**Mining Quality Phrases from Massive Text Corpora**](http://jialu.cs.illinois.edu/paper/sigmod2015-liu.pdf)”, Proc. of 2015 ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'15), Melbourne, Australia, May 2015. (\* equally contributed, [slides](http://jialu.cs.illinois.edu/paper/sigmod2015-liu-slides.pdf))
* SegPhrase Setup: <https://github.com/shangjingbo1226/SegPhrase/issues/4>