What Even Is This Shit?

Basically we are making a program that takes in what type of business the user is looking for, along with qualities of that type of business that are important to the user, and outputs “recommended” businesses that fit the user.

An example of running our app from the command line might look like:

```
>$ our_program me.yml
```

me.yml -

```
category: Pharmacy
lookFor:
     - aspect: ~~fast~~ speed
       weight: 30
     - aspect: ~~courteous~~ service
       weight: 40
```

In this example, we would apply a 30% weight to positive aspect “fast”, a 40% weight to positive aspect “courteous” and the remaining 30% would be applied to the general review.

Maybe we would also have (in me.yml)…

```
avoid:
     - aspect: ~~smelly~~ smell
       weight: 10
```

… if we’re feeling ambitious (so far… not really).

How Are We Going To Do This Shit?

Steps:

1. Prepare data for processing
2. Extract Aspect terms
3. Sentiment analysis for each aspect

Prepare Data for Processing

- Do all of this prep, below, with an “initialize” function.
    - The function saves information in files that are easily computer-readable
    - The function is smart. It knows if “initialize” has been called already, and, if so, will skip all this work.
- Use [Naive Bayes Classifier?] to classify positive and negative words from all 1-star and 5-star reviews.

    - Do this processing ahead of time and save the information in a binary file for easy access
- Take bulk list of reviews and organize it into files by business “category”
    - The meta file has a list of all businesses, each one has entries in “category” (it’s a list)
    - So make files that are like this example:

pharmacy.json

```
{
     “name”:”Walgreens”,
     “gmap_id”: 12345,
     “reviews” : [
          {“rating”:5, “text”:”Good selection!”}
          {“rating”:1, “text”:”Too slow.”}
          {“rating”:2, “text”:”This shit is mid tbh”}
     ]
}
{
     “name”:”CVS”,
     “gmap_id”:54321,
     “reviews” : [
          {“rating”:5, “text”:”Way better than Walgreens.”}
etc...
```

IMPORTANT note about this example: This is formatted in this document so it’s nice and readable for humans. Remember the real files have to be parsed by a computer. So while you are making your writer function, test it with a reader function at the same time, and don’t test it with your eyeballs. :)

Extract Aspect Terms

- Along with category, user will enter aspects – nouns only – they are “looking for” (good) [or “avoiding” (bad)](maybe)
- Extract the aspects from the reviews in the correct category file
    - user noun + top [10?] similar words (word2vec has downloadable trained models for this)
- For each aspect, make a list of reviews that contain the aspect

Analyze Sentiment

Work in progress

Apply Analysis

So for each aspect, you can’t rely on just the rating to tell you how good/bad the aspect is. It’s possible for a user to leave a 2-star rating saying something like “This would be 1-star if it weren’t for [aspect]”, which means the aspect was positive. I figure, for each review…

- Some weight is given to rating (business sentiment)
- Some weight is given to aspect sentiment for each aspect

How much weight is given to each can be tuned by the user for each aspect, but it has to add up to <100%.

We apply a weighted average and give each business a “score”. The top [some number] scoring businesses in the category will be outputted to the user as recommendations, along with their “scores” (or maybe ranked if the scores look bad idk)