Topic Modeling of Yelp Academy Dataset

Yading Guo

2016/11/15

Topic modeling was carried out in this report on Yelp academy dataset. There are two main parts of this report.

- **Topic modeling of all the reviews in the dataset**
- **Comparison of topics of reviews from different sex**

Topic modeling of all the reviews in the dataset

Three major steps are involved in this part.

- Processing the raw reviews into words
- Constructing TDIDF vector used for topic modeling
- Fitting a LDA topic model

Firstly, processing the raw reviews into words

In the beginning, I import some modules for later usage.

- nltk and string module are used for natural languange processing
- json and io are used for file read and write
- graphlab is used for topic modeling
- pyLDAvis is used for topic visualization

In [1]:

```
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
import string
import json
import graphlab
import io
import sexmachine.detector as gender
import pyLDAvis
import pyLDAvis.graphlab
```

```
This non-commercial license of GraphLab Create for academic use is assigned to adenguo@gmail.com and will expire on August 05,
```

```
[INFO] graphlab.cython.cy_server: GraphLab Create v2.1 started. Logging: /tmp/graphlab_server_1479217598.log
```

Some constants were defined and loaded. They are raw data, hyperparameter of constructing TDIDF model, stopwords list and so on.

And some functions are defined.

In [2]:

```
male_id = set()
female_id = set()
data_user = []
data_review = []
with open('data/yelp_academic_dataset_review.json') as f:
    for line in f:
        data_review.append(json.loads(line))
with open('data/yelp_academic_dataset_user.json') as f:
    for line in f:
        data_user.append(json.loads(line))
trim_value = 2
min_length = 10
extra_words = set(["food", "it", "get", "go", "u"])
stop = set(stopwords.words('english'))
exclude = set(string.punctuation)
lemma = WordNetLemmatizer()
detector1 = gender.Detector()
pyLDAvis.enable_notebook()
def write_text_lists(texts_list, file_name):
    f = io.open(file_name, 'w+', encoding='utf8')
    for line in texts_list:
        f.write(u','.join(line) + '\n')
    f.close()
def load_text_lists(file_name):
    f = io.open(file_name, 'r', encoding='utf8')
    lines = f.readlines()
    return [line.strip().split(',') for line in lines]
def clean(doc):
    stop_free = " ".join([i for i in doc.lower().split() if i not in stop])
    punc_free = ''.join(ch for ch in stop_free if ch not in exclude)
    normalized = " ".join(lemma.lemmatize(word) for word in punc_free.split())
    return normalized
def data_review_to_sf_bag_words(texts_list):
    sf_text = graphlab.SFrame({'text': texts_list})
    encoder = graphlab.feature_engineering.WordCounter()
    transformed_sf = encoder.fit_transform(sf_text)
    return transformed_sf
def data_review_to_sf_tfidf(texts_list):
    sf_text = graphlab.SFrame({'text': texts_list})
    encoder = graphlab.feature_engineering.WordCounter()
    bag_words_sf = encoder.fit_transform(sf_text)
    encoder_tfidf = graphlab.feature_engineering.TFIDF('text')
    encoder_tfidf = encoder_tfidf.fit(bag_words_sf)
    result = encoder_tfidf.transform(bag_words_sf)
    return result
def create_topic_model(text_file):
    print "loading save text"
    text_list = load_text_lists(text_file)
    print "creating bag of words vector"
    bag_of_words_vector = data_review_to_sf_bag_words(text_list)
    print "triming vector by value "+ str(trim_value)
    bag_of_words_vector = bag_of_words_vector['text'].dict_trim_by_values(trim_value)
    print "delect short line then " + str(min_length)
    ix = bag_of_words_vector.apply(lambda x: len(x.keys()) >= min_length)
    bag_of_words_vector = bag_of_words_vector[ix]
    print "remove extra words"
    bag_of_words_vector = bag_of_words_vector.dict_trim_by_keys(extra_words,exclude=True)
    print "creating tfidf vector"
    tfidf_vector = graphlab.text_analytics.tf_idf(bag_of_words_vector)
    model = graphlab.topic_model.create(tfidf_vector,
                              num_topics=10,       # number of topics
                              num_iterations=100,   # algorithm parameters
                              alpha=10, beta=0.1)  # hyperparameters
    return model,tfidf_vector
def split_male_female(data_user, data_review):
    for user in data_user:
        if detector1.get_gender(user['name']) == "male":
            male_id.add(user['user_id'])
        elif detector1.get_gender(user['name']) == "female":
            female_id.add(user['user_id'])
        else:
            pass
    male_review = []
    female_review = []
    for review in data_review:
        if review['user_id'] in male_id:
            male_review.append(review['text'])
        elif review['user_id'] in female_id:
            female_review.append(review['text'])
        else:
            pass
    return male_review,female_review
```

Following steps were done when the raw data is processed into list of words.

1. Stop words are removed.
2. Punctuations are removed.
3. Lemmatization are performed on each words.
4. Tokenization the text into a list of words.

In the following funtions, the function *clean* is the workhorse of there steps. There processed data is then write into a file using function *write_text_lists*

In [3]:

```
text_data = [x['text'] for x in data_review]
text_list1 = [clean(doc).split() for doc in text_data]
write_text_lists(text_list1, 'clean_text.txt')
```

Secondly, construction of TFIDF vector

There are two procedures in this part.

1. Removing very rare words which is the words appearing less than 2 times in the corpus. And delete very short reviews which is reviews which contain less than 10 ur
2. A vector of bag of words is construct and then it is converted into a vector of TFIDF.

The first part of function *creat_topic_model* is doing above two steps.

Thirdly, Fitting a LDA topic model

The second part of funtion *creating_topic_model* is doing this part. A LDA model was constructed. The interative visualization of the topic model is construct.

In [4]:

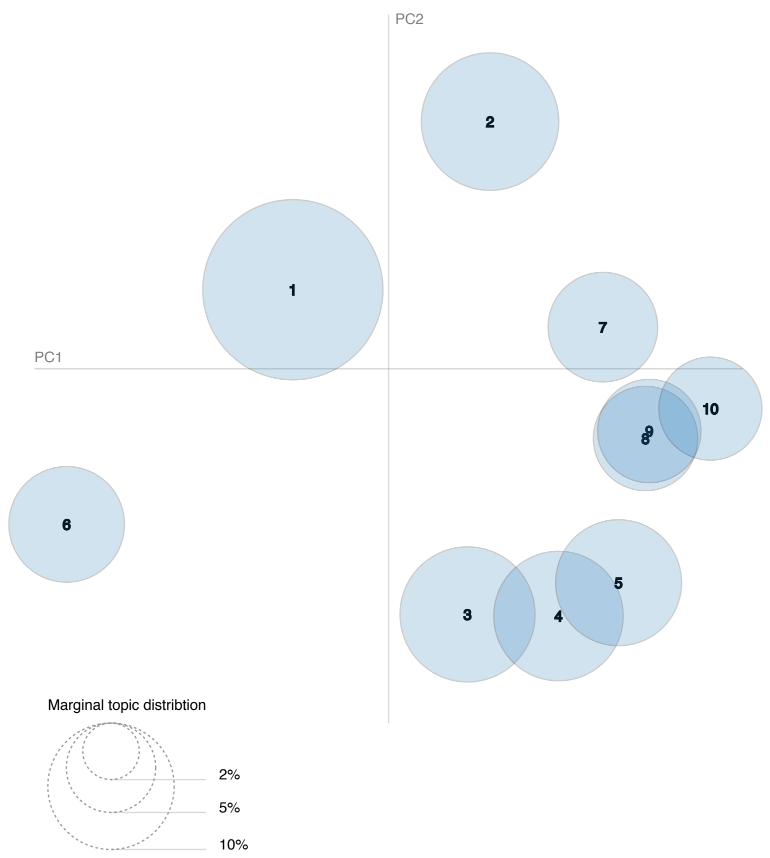
```
all_model,all_tfidf = create_topic_model('clean_text.txt')
pyLDAvis.graphlab.prepare(all_model, all_tfidf)
```

```
loading save text
creating bag of words vector
triming vector by value 2
delect short line then 10
remove extra words
creating tfidf vector

Learning a topic model

       Number of documents     312809

           Vocabulary size      53821

   Running collapsed Gibbs sampling

+-----------+---------------+----------------+-----------------+

| Iteration | Elapsed Time  | Tokens/Second  | Est. Perplexity |

+-----------+---------------+----------------+-----------------+

| 10        | 5.11s         | 1.23437e+07    | 0               |

| 20        | 10.51s        | 1.19085e+07    | 0               |

| 30        | 15.31s        | 1.31026e+07    | 0               |

| 40        | 19.97s        | 1.3434e+07     | 0               |

| 50        | 24.52s        | 1.36925e+07    | 0               |

| 60        | 29.00s        | 1.37114e+07    | 0               |

| 70        | 33.59s        | 1.2918e+07     | 0               |

| 80        | 38.09s        | 1.31997e+07    | 0               |

| 90        | 42.76s        | 1.25602e+07    | 0               |

| 100       | 47.48s        | 1.29524e+07    | 0               |

+-----------+---------------+----------------+-----------------+

Out[4]:
```

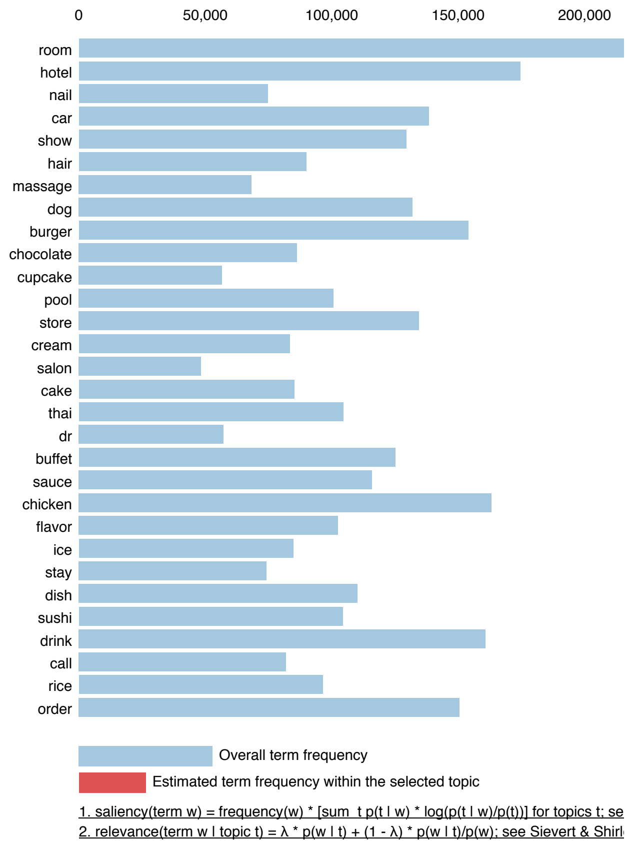
Selected Topic: 0 Previous Topic Next Topic Clear Topic

Slide to adjust relevance metric:(2)
λ = 1
0.0 0.2 0.4

Intertopic Distance Map (via multidimensional scaling)



Top-30 Most Salient Terms[1]



As we can see from the visualization. There are several apparent topic. For example, topic 8 is about foreign cuisine. Topic 3 is about shops providing service other than f
Topic 1 is about ordinary food.

Comparison of topics of reviews from different sex

I split the reviews into reviews from male and reviews from female.

This is done by function *split_male_female*.

Then I do exactly the same procedures as the first part of this report on both reviews to produce male topic model and female topic model and visualized them to compar

In [5]:

```
male_review, female_review = split_male_female(data_user, data_review)
male_text_list = [clean(doc).split() for doc in male_review]
female_text_list = [clean(doc).split() for doc in female_review]
write_text_lists(male_text_list, 'male_clean_text.txt')
write_text_lists(female_text_list, 'female_clean_text.txt')
```

In [6]:

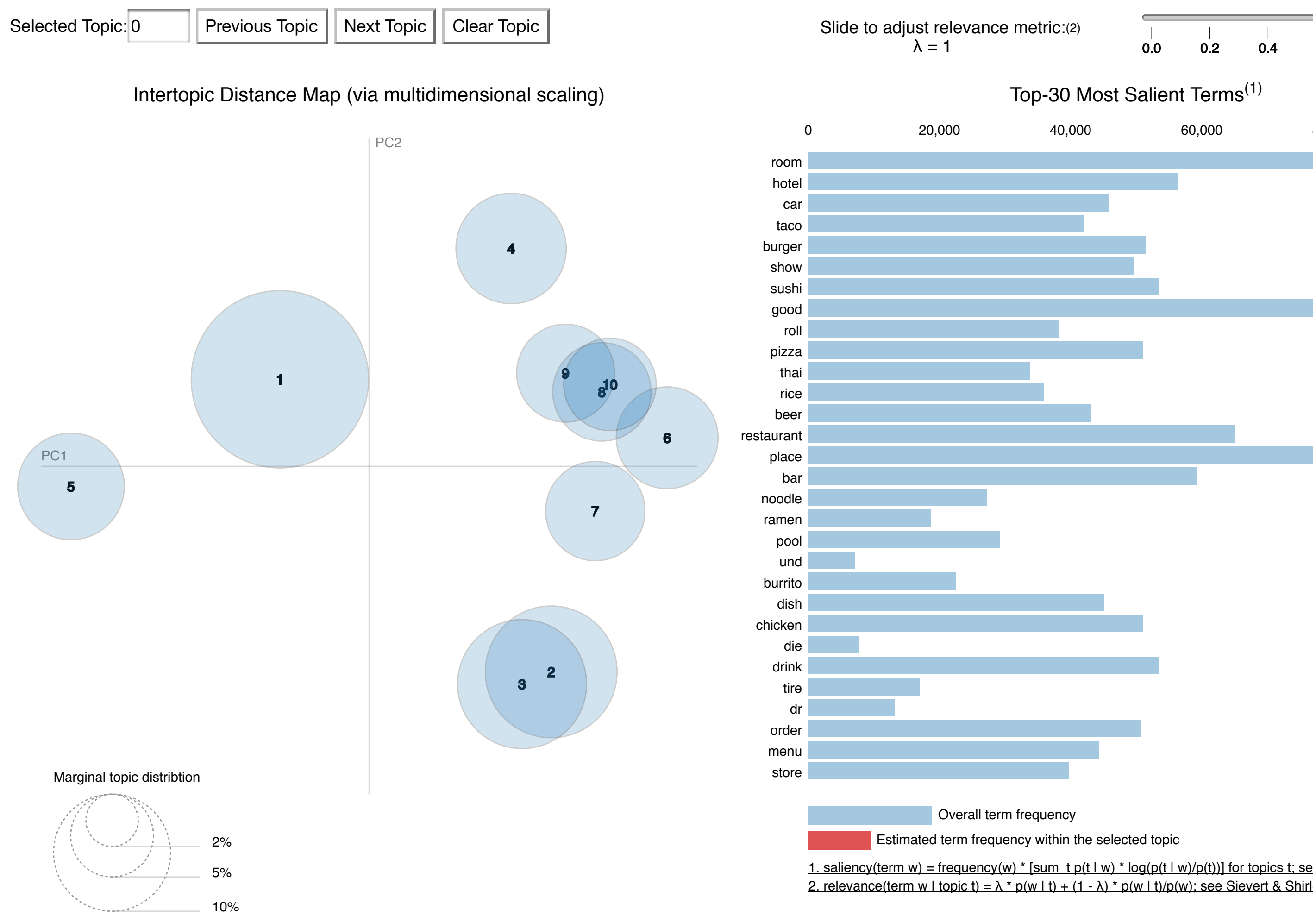
```
male_model,male_tfidf = create_topic_model('male_clean_text.txt')
female_model,female_tfidf = create_topic_model('female_clean_text.txt')
```

```
loading save text
creating bag of words vector
triming vector by value 2
delect short line then 10
remove extra words
creating tfidf vector

Learning a topic model

       Number of documents    100570

           Vocabulary size     33456

   Running collapsed Gibbs sampling

+-----------+---------------+----------------+-----------------+

| Iteration | Elapsed Time  | Tokens/Second  | Est. Perplexity |

+-----------+---------------+----------------+-----------------+

| 10        | 2.01s         | 1.04521e+07    | 0               |

| 20        | 3.95s         | 1.02329e+07    | 0               |

| 30        | 5.90s         | 9.68014e+06    | 0               |

| 40        | 7.83s         | 1.11755e+07    | 0               |

| 50        | 9.67s         | 1.13666e+07    | 0               |

| 60        | 11.54s        | 9.9035e+06     | 0               |

| 70        | 13.48s        | 1.032e+07      | 0               |

| 80        | 15.37s        | 1.05846e+07    | 0               |

| 90        | 17.36s        | 9.75412e+06    | 0               |

| 100       | 19.30s        | 1.06694e+07    | 0               |

+-----------+---------------+----------------+-----------------+

loading save text
creating bag of words vector
triming vector by value 2
delect short line then 10
remove extra words
creating tfidf vector

Learning a topic model

       Number of documents    127307

           Vocabulary size     34166

   Running collapsed Gibbs sampling

+-----------+---------------+----------------+-----------------+

| Iteration | Elapsed Time  | Tokens/Second  | Est. Perplexity |

+-----------+---------------+----------------+-----------------+

| 10        | 2.47s         | 9.77887e+06    | 0               |

| 20        | 4.89s         | 1.04131e+07    | 0               |

| 30        | 7.24s         | 1.05262e+07    | 0               |

| 40        | 9.58s         | 1.065e+07      | 0               |

| 50        | 11.85s        | 1.12204e+07    | 0               |

| 60        | 14.12s        | 1.07879e+07    | 0               |

| 70        | 16.35s        | 1.17748e+07    | 0               |

| 80        | 18.59s        | 1.06742e+07    | 0               |

| 90        | 20.85s        | 1.10036e+07    | 0               |

| 100       | 23.23s        | 9.9042e+06     | 0               |

+-----------+---------------+----------------+-----------------+
```

In [7]:

```
pyLDAvis.graphlab.prepare(male_model, male_tfidf)
```

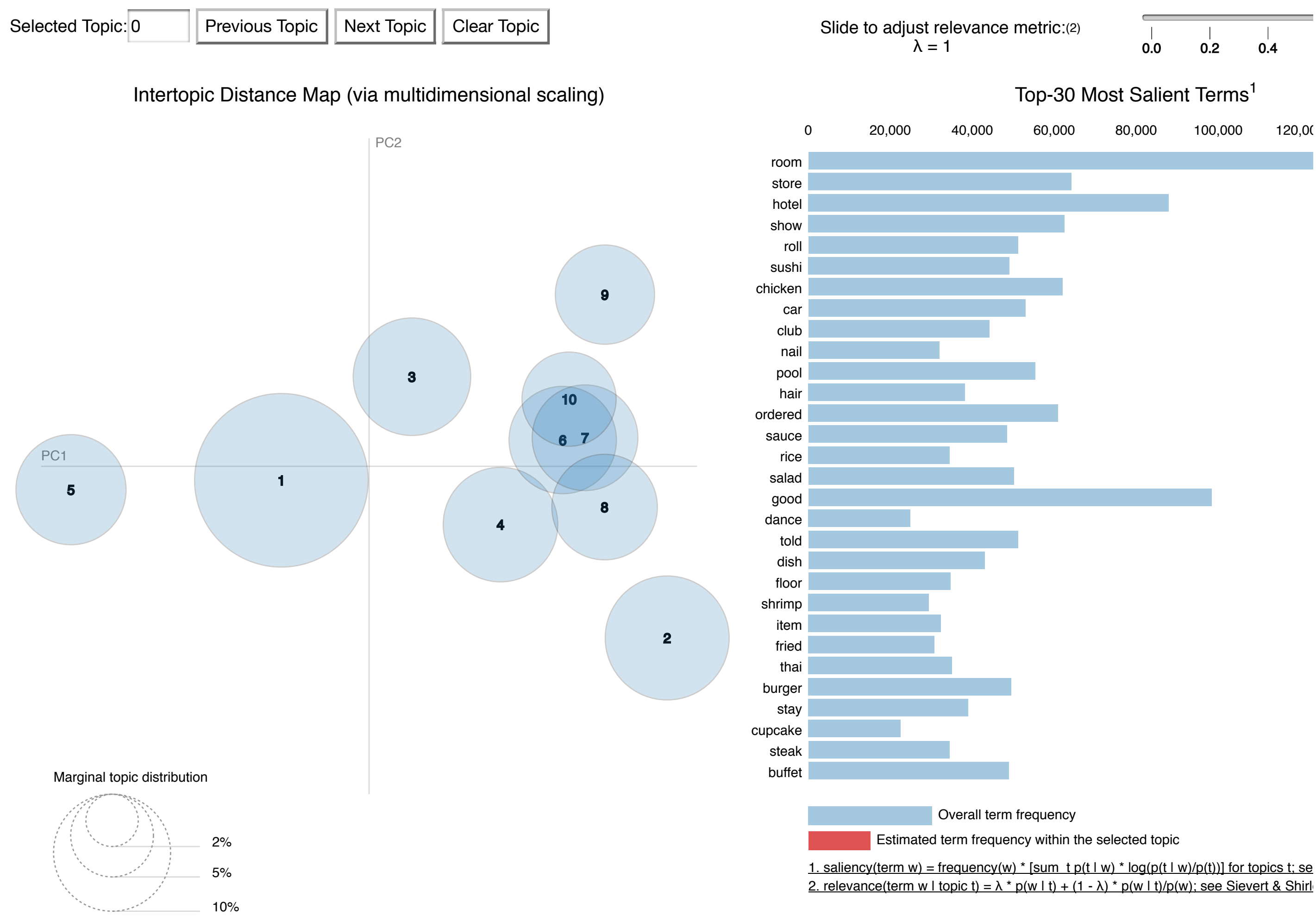
Out[7]:



In [8]:

```
pyLDAvis.graphlab.prepare(female_model, female_tfidf)
```

Out[8]:



As we can see, there are obvious difference existing between male's and female's topics. There is a topic, 3, for male is about car, repair and problem. Other topics from r gambling(topic 4) and shows(topic 6). These topics don't show themselves in female's topics. Female's topics are concentrate on drink and dessert(topic 9), hair cut and i 5).