CS410 Final Project Report

Team American River: daechoi2, dedobbe2, ss117

1. Code Function

The code's function is to create a generative model using LARA (Latent Aspect Rating Analysis) in research paper "Latent aspect rating analysis without aspect keyword supervision" by Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011. As a part of this process, we are inferring both opinion ratings on topical aspects (e.g., location, service of a hotel or prodcut) and the relative weights reviewers have placed on each aspect based on review content and the associated overall ratings.

One of the key advantages of this code is that it doesn't require the assumption of pre-specified aspects by keywords. As a part of the functionality, we are simultaneously mining:

- Latent topical aspects
- Ratings on each identified aspect
- Weights placed on different aspects by a reviewer

Thus, this function can be used for a wide variety of interesting application tasks, such as aspect-based opinion summarization, personalized entity ranking and recommendation, and reviewer behavior analysis.

We will be investigating this model over two different data sets:

- Hotel reviews from TripAdvisor (in the folder *Data/HotelData*)
- Product reviews of MP3 players from Amazon (in the folder *Data/ProductData*)

2. Code Implementation

We implemented data curation scripts for raw data in Data/*/RawData:

- DataCleaning/HotelDataCleaner.py
- DataCleaning/ProductDataCleaner.py

These data curation scripts

(1) Remove the reviews with any missing aspect rating or document length less than 50 words (to keep the content coverage of all possible aspects)
(2) Convert all the words into lower cases
(3) Remove punctuations, stop words, and the terms occurring in less than 10 reviews in the collection

We implemented the code in three main files in the folder *Analysis*:

- *HotelAnalysis.py*
  Parse the hotel review data and create a list of vocabularies with frequency

- *ProductAnalysis.py*
  Parse the Amazon product review data and create a list of vocabularies with frequency

- *AnalyzeMethods.py*
  Include main algorithm functions and utility functions shared by Hotel Analysis and Product Analysis

We have two helper functions:

1. *parseWordsForSentence()*
   We parse words for a sentence, and use stop words and tokenizers to get the associated terms

2. *parseWords()*
   We parse words for a review, and use stop words and tokenizers to get the associated terms

The flow of the main functions is as follows:

1. *genStopwords()*
   Generate stopwords and fetch it from *Data/StopWords.json*

2. *getData()*
   Fetch data from *Data/HotelData/CleanData* or *Data/ProductData/CleanData*

3. *createVocab()*
   Go through each review, use parseWords to get the stemmed terms, get their frequency in the review and collect them in separate lists.

4. *runAlgorithm()*
   Go through the entire algorithm here. The functions here include:
   a) *generateAspectParameters()*
      As a part of this, we generate aspect parameters for each review. We do this by running an iterative Expectation Maximization algorithm over multiple iterations to get the mean and standard deviation of aspect parameters. Expectation Maximization happens by the following:
      1. initializeParameters()
         Initialize the different parameters needed for the EM algorithm and calculate an initial value of mean and standard deviation.
      2. *Estep()*
         For each review in the corpus, use the parameters to infer the mean and standard deviation of the aspect parameters, while calculating the likelihood.

3. *Mstep()*
   Keep updating the parameters and means and standard deviations to maximize the parameters that are found in the E step.

b) *sentenceLabeling()*
   Use the means and standard deviation derived for the aspect parameters to update a list of labels for each review, where the values will be -1 or 1, with 1 being associated with the word having the highest aspect rating.

c) *createWordMatrix()*
   Generate a word matrix, where we go through each review and generate a matrix in which we are multiplying the word frequency with the review label to generate the sentiment analysis for that row.

3. How to Use the Software

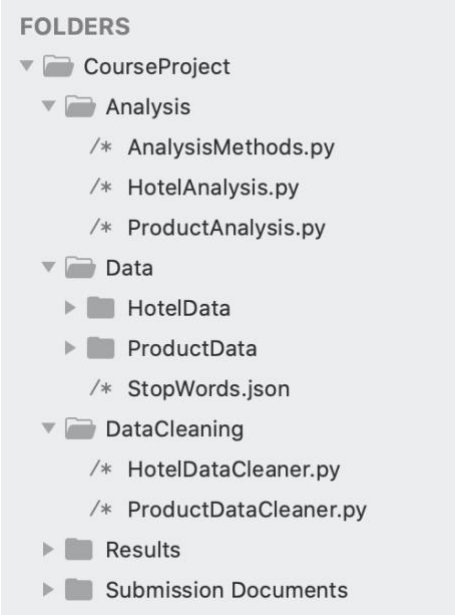
We used a Macbook Pro with a 2.9 GHz Intel Core i9 processor and 2400 MHz and 32GB memory to produce the following results. Processing 100 file items (3508 reviews) took about 6 - 10 hours to produce the results described in the following Section 4 Full Data Results. To complete a test in more reasonable time, we created smaller sets of data in the folder *Data/*/testData/*.

The following describes the steps to reproduce results with the small data set. You can follow along with the video https://mediaspace.illinois.edu/media/1_nuiybt07.

1. Download the git repository (git clone https://github.com/JackDeDobb/CourseProject)

```
maximuschoi@Maximus-Choi-2 CS410_Text_Info_System % git clone https://github.com/JackDeDobb/CourseProject.git
Cloning into 'CourseProject'...
remote: Enumerating objects: 45590, done.
remote: Total 45590 (delta 0), reused 0 (delta 0), pack-reused 45590
Receiving objects: 100% (45590/45590), 840.45 MiB | 22.27 MiB/s, done.
Resolving deltas: 100% (23690/23690), done.
Updating files: 100% (23382/23382), done.
maximuschoi@Maximus-Choi-2 CS410_Text_Info_System % cd CourseProject
```

The repository has the following structure



2. Go to the folder *CourseProject/Analysis* and turn the following command to produce the results with the small data set of **TripAdvisor Hotel**.

   python HotelAnalysis.py

```
maximuschoi@Maximus-Choi-2 CourseProject % cd Analysis
maximuschoi@Maximus-Choi-2 Analysis % ls
AnalysisMethods.py      HotelAnalysis.py        ProductAnalysis.py
maximuschoi@Maximus-Choi-2 Analysis % python HotelAnalysis.py
```

   The analysis will run three iterations and produced the following files with results in the folder *Results:*

   - *TestSet_HotelFinalResults.txt*
   - *TestSet_HotelFinalResultsStats.json*

   *Note: Passing parameter “full” after “python HotelAnalysis.py: will run the script on all hotel reviews currently in the clean data folder rather than the test data folder.

3. Go to the folder *CourseProject/Analysis* and turn the following command to produce the results with the small data set of **Amazon Product**.

   python ProductAnalysis.py

```
maximuschoi@Maximus-Choi-2 Analysis % ls
AnalysisMethods.py      HotelAnalysis.py        ProductAnalysis.py
maximuschoi@Maximus-Choi-2 Analysis % python ProductAnalysis.py
```

   The analysis will run three iterations and produced the following files with results in the folder *Results:*

- *TestSet_ProductFinalResults.txt*
- *TestSet_ProductFinalResultsStats.json*

*Note: Passing parameter “full” after “python ProductAnalysis.py: will run the script on all product reviews currently in the clean data folder rather than the test data folder.

4. Full Data Results

Model Description

In the models of LRR (Latent Rating Regression) and LDA, the aspects must be given through keywords specified by users, restricting its usefulness in applications. Such supervision of specifying aspects with keywords requires manual work and is not always available. More importantly, in many cases, it is often unclear what are the aspects actually commented on in the reviews, thus it is very difficult, if not impossible, to pre-specify the aspects beforehand.

This model utilizes a generative Latent Aspect Rating Analysis Model proposed in the paper to identify

1. latent topical aspects,
2. ratings on each identified aspect, and
3. weights placed on different aspects by a reviewer.

It is assumed that the text content describing a particular aspect is generated by sampling words from a topic model (i.e., a multinomial word distribution) corresponding to the latent aspect, the latent rating on an aspect is determined based on the words describing each aspect with latent sentiment polarities, and the overall rating is generated based on a weighted combination of aspect ratings where the (latent) weights reflect the relative emphasis on each aspect by the reviewer. Thus, this is an extension of LRR model to perform both aspect segmentation and aspect rating prediction in a unified framework.

Expected results

On running the code, you will see two JSON files added to the folder *Results* for each datasets - *TestSet_HotelFinalResultsStats.json* and *TestSet_ProductFinalResultsStats.json*

In them you will find the data for

1. Statistics of data sets

This includes the total number of items, reviews, reviewers, average length of a review utilized in the model. You will also see the mean and standard deviation of the review ratings from the review data used. You can check them out for the full data tested in Table 1 below.

2. Topical aspects learned on reviews

As described above, as the model does automatic identification of latent topical aspects, we have generated the lists of top words of the highest generation probability which can be considered as contributing most to the overall ratings, in both a positive and negative sense.

These lists are captured in the json as High Overall Ratings and Low Overall Ratings. They are also shown for the full data in Table 2 and 3

3. Aspect rating prediction performance on Reviews

As the model is also using the generated aspects to predict ratings, we are showing the statistics of

(1) Mean Square Error of the predicted aspect ratings compared with the ground-truth aspect ratings; (2) Pearson correlation inside reviews measures how well the predicted aspect ratings can preserve the relative order of aspects within a review given by their ground-truth ratings.

This is shown as Total MSE and Total Pearson in the json and in Table 4 for the full data set.

Full set Results

The following tables show results from running analysis on the full data sets. We successfully created a generative model using LARA.

Table 1: Statistics of data sets

	#Item	#Review	#Reviewer	Avg Len	Rating
Hotel	100	3508	3447	448.09	3.845 ±1.194
Products	100	9020	8640	542.94	3.851 ±1.315

Table 2: Topical aspects learned on Hotel reviews

Low overall Ratings			High Overall Ratings		
hotel	stay	nice	stay	hotel	staff
night	staff	locat	clean	locat	breakfast
pool	time	hollywood	time	night	bed
clean	walk	resort	help	pool	friendli
breakfast	day	bread	comfort	nice	restaur
check	servic	look	servic	desk	day
park	book	airport	park	recommend	walk
properti	hilton	love	look	price	free
free	branson	front	front	close	lot
la	motel	close	didnt	check	travel

Table 3: Topical aspects learned on Product reviews

Low overall Ratings			High Overall Ratings		
jack	headphon	warranti	player	ipod	music
service	screen	song	vision	screen	video
devic	batteri	charge	love	great	headphon
player	usb	scratch	qualiti	incl	aapl
service	time	wall	buy	sound	send
back	watch	movi	highli	volume	hour
unit	product	zen	photo	easi	review
look	purchas	bought	space	bud	tv
creativ	comput	file	protect	file	hand
reset	hour	softwar	easi	gig	day

Table 4: Aspect rating prediction performance on reviews

	LDA+LRR	sLDA+LRR	LARAM (paper)	LARAM(hotel reviews for our model)	LARAM (MP3 reviews for our model)
MSE	2.130	2.360	1.234	10.814	9.799
P (Pearson coefficient)	0.080	0.079	0.228	-0.103	0.078

Conclusion

We can see that while our model doesn't match the results of the paper exactly, we do see a trend of results improving as the test set of reviews keep increasing. We can also see that we are generating topical aspects which make sense- for example, our Products high rating aspects includes a lot of positive rating associated with the word 'player' and negative rating associated with 'jack'.

We are confident with some tuning of our model, and changes in algorithm to incorporate multiprocessing, we will achieve similar or even better results compared to the model in the paper.

5. Project Team Members Contributions

- Maximus Choi (daechoi2@illinois.edu)
- Jackson DeDobbelaere (dedobbe2@illinois.edu) – Captain
- Subramanian Shankar (ss117@illinois.edu)

The team members together reviewed the research paper to reproduce, analyzed data, and designed, built, and tested the code. Subramanian focused on building the base code frame and algorithms. Jackson and Maximus focused on curating the data and improving the code. The team members together validated the results and built the report documentation.