9_DATA_ANALYTICS

Hotels Review Reviewed

Sentiment Analysis for Hotel Reviews

User reviews and comments on hotels on the web are an important information source in travel planning. We present a system that collects such reviews from the web and creates particular rating based on location, food, rooms, service, value and overall rating to create classified and structured information. The task of rating the hotels by the reviews is done by performing sentiment analysis on reviews. We applied feature-aspect based approach to the evaluation.

Index

Topic	Page Number
Abstract	2
Introduction	2
Methodology	4
Data Acquisition system	4
Aspect Based Sentiment Analysis	6
Sentiment Tree Bank	6
Recursive Neural Tensor Network	8
Breaking of reviews into sentences	9
Score methodology and normalisation	9
Feature-aspect based analysis	11
User Interface	13
Logic	15
Evaluation	17
Conclusion	19
Future Plans	19
Annexure	20
Documentation	20
Attachments	21
References	22

I. ABSTRACT

User reviews and comments on hotels on the web are an important information source in travel planning. We present a system that collects such reviews from the web and creates particular rating based on location, food, rooms, service, value and overall rating to create classified and structured information. The task of rating the hotels by the reviews is done by performing sentiment analysis on reviews. We applied feature-aspect based approach to the evaluation.

Keywords: Hotel reviews, Text Mining, Sentiment Analysis

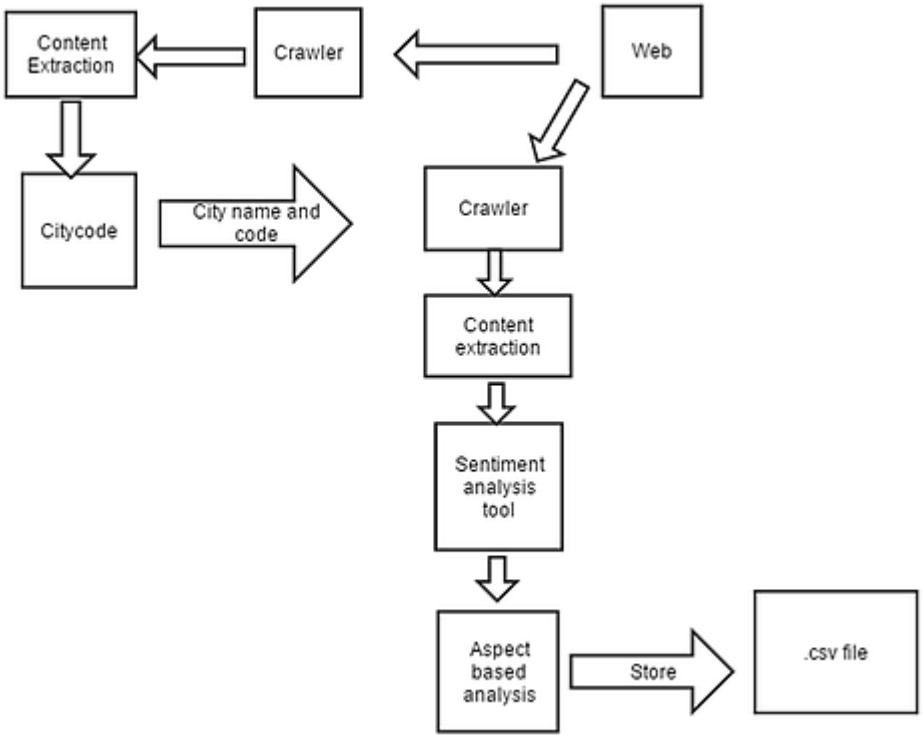
II. INTRODUCTION

The perceived value of reviews on the Web is uncontested: consumer surveys show that people cite product reviews as a top influencer in purchase decisions. According to Nielsen, consumer recommendations are the most credible form of advertising among 78% of survey responders (Survey 2007); and a BIGresearch survey indicates that 43.7% of consumer electronics purchases are affected by word of mouth. Given the important influence of reviews, we might then ask, is it possible to extract a score from a collection of reviews that accurately reflects the relative quality of the hotels under review?

Our goal in this work is to address this issue and rank hotels in particular city accordingly. So, we collected reviews of hotels from sites like tripadvisor.in, holidayiq.com and social networking website, twitter using web crawler scrapy and then performed aspect based sentiment analysis on these reviews using Stanford CoreNLP package.

Scrapy is an application framework for writing web spiders that crawl web sites and extract data from them. We used it to extract hotel names, their reviews and description given the city name. The information for then presented in JSON format for further sentiment analysis.

Sentiment Analysis, which is used by the package given by Stanford's CoreNLP is a part of natural language processing. The review string is parsed in tree form and using grammar and dictionary, appropriate analysis score is given to reviews.



It is also worth mentioning that human classification has around 70% correctness because human raters typically agree about 70% of the time. Thus, a system that has around 70% accuracy is as good as human raters, even though it may not sound too impressive. If a program were "right" 100% of the time, the average human would still disagree with it around 30% of the time.

The information was extracted in both JSON and CSV format. The information was then made available for use through a website.

We will give an overview of the system in Section III and discuss the major components in more detail, the data acquisition from the web (Section III-I), the sentiment analysis (Section III-II) and other key methods in subsequent sub-sections of section III. The user interface will be presented in Section IV. In Section V evaluation results for the analysis system will be presented. The report is then concluded with a brief mention of future plans.

III. METHADOLOGY

The methodology used for the task could be broken down into three steps- data acquisition, sentiment analysis, and user-interface.

III.I **Data Acquisition System**

The acquisition of reviews from the web is handled by a web crawler scrapy.

Crawling is quite simple at its core:

1. Select a URL to crawl
2. Fetch and parse page
3. Save the important content
4. Extract URLs from page
5. Add URLs to queue
6. Repeat

Scrapy is a Python package that aims at easy, fast, and automated web crawling. Scrapy uses a class called Item as a container for the crawled data.

The class that actually does the crawling is called Spider (for obvious reasons). We feed the spider with a list of starting URLs. The spider goes to each of the URL, extracts data that is desired, and stores them as a list of instances of the class that we previously defined.

Scraping starts at expedia.co.in, from where the crawler retrieves city name and the code (this is the code that uniquely identifies a city, used later). Then using the city name and code for a particular city (say New Delhi) second spider defines for each city a set of crawl configurations that define a start URL, URL patterns for links to follow, target URL patterns for pages containing reviews. All the URLs usually point to dynamic web pages, that is, the content of the web pages can change between visits. Also, the web pages most times contain hundreds of links, most of them being irrelevant for retrieving reviews (e.g. advertisements, other hotels, etc). The distinction between links to follow and target pages is required because the crawler often has to go through several intermediate pages to get at the review pages. The calls are made

asynchronously. When a target page is retrieved a content extraction module is applied that extracts the relevant textual content of the review but also other metadata such as price, amenities, star rating, location. So, our output of this part is JSON file containing hotel name, location, price, star rating, amenities, and reviews. This file is passed to the analysis system. The review texts there first split into text segments that become the units of further analysis.


```
{"city": "Kausani", "code": "d6129774"},
{"city": "Khajuraho", "code": "d1470"},
{"city": "Lonavala", "code": "d8681"},
{"city": "Lucknow", "code": "d2091"},
{"city": "Ludhiana", "code": "d2158"},
{"city": "Madhapur", "code": "d6130337"},
{"city": "New Delhi", "code": "d177865"},
{"city": "New Friends Colony", "code": "d6239243"},
{"city": "Nilambur", "code": "d6136982"},
{"city": "Nilgiri Hills", "code": "d6053323"},
{"city": "Noida", "code": "d6003751"},
{"city": "North Goa", "code": "d6138899"},
{"city": "Ganpatipule", "code": "d6138355"},
{"city": "Garjia", "code": "d6139762"},
```


City *Name*


```
{"title": "Clark Heights", "holidayiqR": ["\ufffdI found this hotel batter than any
{"title": "BnB New Delhi Homestay", "holidayiqR": ["I stayed at BNB New Delhi durin
{"title": "Avalon Courtyard", "holidayiqR": ["Service was very friendly and nice.
{"title": "Hotel Citi International", "holidayiqR": ["It was a good experience. It
{"title": "Hotel Chand Palace", "holidayiqR": ["My overall experience wsa good in t
{"title": "Astoria", "holidayiqR": ["Nice location, hotel appearance is simple. Foo
{"title": "Hotel City Centre", "holidayiqR": ["My family and I stayed at the hotel
{"title": "Amby Inn", "holidayiqR": ["Awesome experience, just loved the place as I
{"title": "Clarion Collection", "holidayiqR": ["A good modern style hotel with nice
{"title": "Amara Hotel", "holidayiqR": ["Very nice hotel structure, the rooms are v
{"title": "Anoop Hotel", "holidayiqR": ["Overall experience at this hotel was avera
{"title": "Hotel Chirag Residency", "holidayiqR": ["The best part is that the hotel
{"title": "Hotel City Park", "holidayiqR": ["City park is one of the best hotel in
{"title": "Hotel Chanakya Inn", "holidayiqR": ["I was in Delhi and next morning I h
{"title": "Basil The Residency", "holidayiqR": ["This Basil The Residency is budget
{"title": "Hotel Broadway", "holidayiqR": ["The building seems to have passed more
{"title": "Hotel Centrum", "holidayiqR": [], "holidayiqU": "http://www.holidayiq.co
{"title": "Hotel Chanchal Deluxe", "holidayiqR": ["Safety, cleanliness, and comfort
```


Hotel Review

III.II **Aspect based Sentiment analysis**

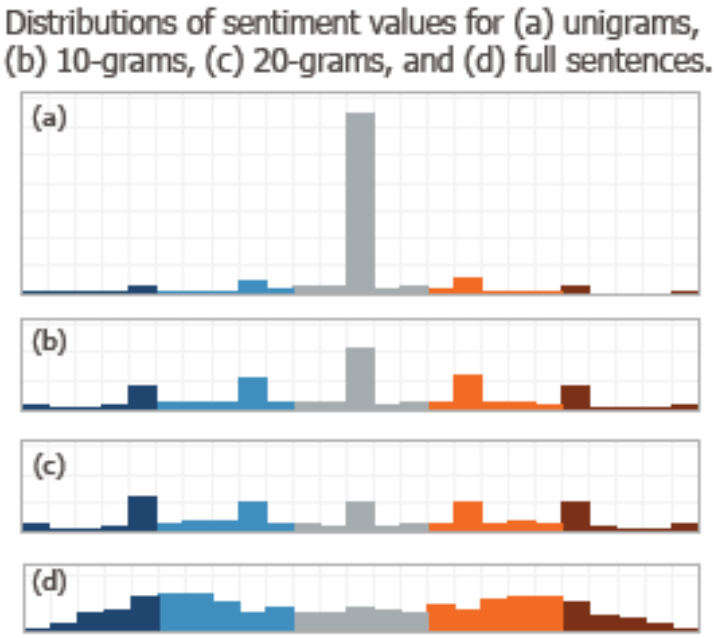
Here we used Stanford's CoreNLP package (in JAVA) which used Recursive Neural Tensor Networks and the Stanford Sentiment Treebank. The combination of new model and data results in a system for single sentence sentiment detection. The strings feed to the model was a single sentence and not a review, i.e. the reviews were divided into sentence and a score was given to each of the sentence for finer classification of views. The output after feeding the string is a number (between 1 and 10) that is then averaged across all reviews for a particular hotel to obtain the score.

Following is the discussion on algorithm used for it and some noteworthy steps.

A. Sentiment Tree Bank

Bag of words classifiers can work well in longer documents by relying on a few words with strong sentiment like 'awesome' or 'exhilarating.' From a linguistic or cognitive standpoint, ignoring word order in the treatment of a semantic task is not plausible, and, as it cannot accurately classify hard examples of negation. Correctly predicting these hard cases is necessary to further improve performance. Sentiment tree bank is used to train the algorithm.

The following is a distribution of sentiment values for phrases of different length and sentences. The noteworthy point is that coreNLP gives the sentiment score to phrase of words rather than single word (however the extremity in sentiment values in phrases of longer length decreases rapidly).



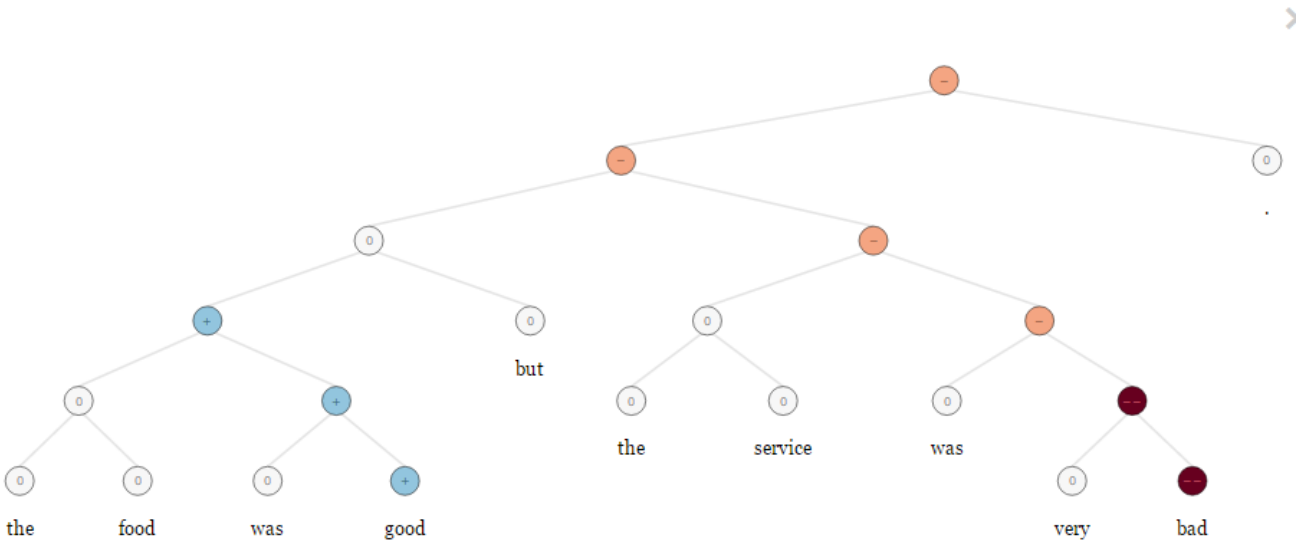
The sentiment pipleline is called using subprocess method of python, giving the review string as standard input (stdin) and calling this function for each review.

```
# print Rarray
for reviewString in Rarray:
    if reviewString:
        # All
        proc = subprocess.Popen(
            'java -cp "*" -mx5g edu.stanford.nlp.sentiment.SentimentPipeline -stdin -output
                probabilities',stdout=subprocess.PIPE,
            stdin=subprocess.PIPE, shell=True)
        proc.stdin.write(reviewString)
        proc.stdin.close()
        result = proc.stdout.read()
```

Stanford Sentiment Treebank includes fine grained sentiment labels for 215,154 phrases in the parse trees of 11,855 sentences and presents new challenges for sentiment compositionality. To address them, we introduce the Recursive Neural Tensor Network.

B. Recursive Neural Tensor Network

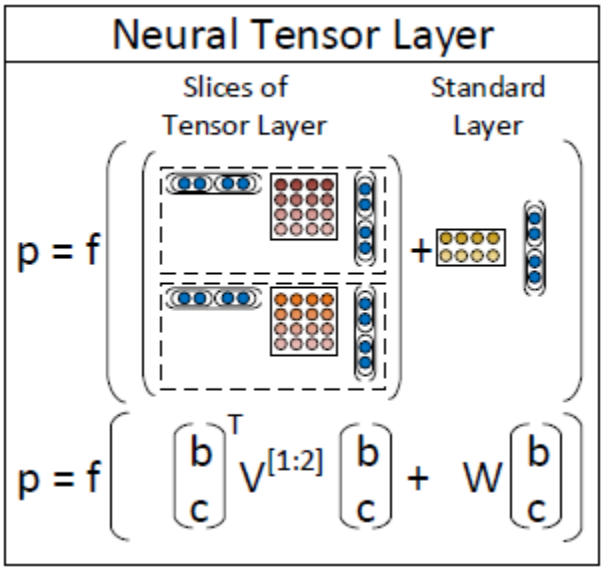
The recursive neural tensor network, commonly known as RNTN is the main backbone of the algorithm. This parses the given string in to sentiment binary tree. For an example,



The string given for above example was “the food was very good but the service was very bad.” (A typical hotel review)

The string is parsed using grammar and dictionary of the coreNLP package. Then score is given to each node of tree recursively, i.e. a lower node affects the higher node. For instance- the adverb “very” adds to the sentiment value of “bad” in the example. The score given to root node is thus the sentiment value for the sentence.

The computation is done using tensor function for all nodes. The following is figure for a single tensor layer-



We define the output of a tensor product h € R^d via the following vectored notation and the equivalent but more detailed notation for each slice $V^{[i]}$ € R^{dxd}:

$$h = \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix}; h_i = \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[i]} \begin{bmatrix} b \\ c \end{bmatrix}$$

Where $V^{[1:d]}$ € $R^{2dx2dxd}$ is the tensor that defines multiple bilinear forms. Each dashed box represents one of d-many slices and can capture a type of influence a child can have on its parent. The RNTN uses this definition for computing p1:

$$p_1 = f\left(\begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix} + W \begin{bmatrix} b \\ c \end{bmatrix}\right)$$

Where W is as defined in the previous models. The next parent vector p2 in the tri-gram will be computed with the same weights:

$$p_2 = f\left(\begin{bmatrix} a \\ p_1 \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} a \\ p_1 \end{bmatrix} + W \begin{bmatrix} a \\ p_1 \end{bmatrix}\right)$$

The recursive models work very well on shorter phrases, where negation and composition are important.

It pushes the state of the art in single sentence positive/negative classification from 80% up to 85.4%. The accuracy of predicting fine-grained sentiment labels for all phrases reaches 80.7%, an improvement of 9.7% over bag of features baselines. Lastly, it is the only model that can accurately capture the effects of negation and its scope at various tree levels for both positive and negative phrases.

C. Breaking of reviews into sentences

Instead of using reviews as the atomic unit for sentiment analysis, we divided the reviews into sentences. This process (dividing of reviews into sentences) was done after a good amount of evaluation.

It was noted that evaluating a text at the document level has some disadvantages. For example, a negative review does not necessarily mean that complete object (hotel in this case) is negative! There can be some specific aspect about that particular object that is positive. Likewise, a positive evaluation does not mean that the author dislikes everything about the object. For instance, in hotel reviews an author usually writes both positive and negative aspects about that hotel, even though the overall sentiment of the review can be either positive or negative. To obtain such detailed aspects, we will have to go to the sentence level and extract the interesting features.

Also, as it was evident from distribution of sentiment value that the accuracy of sentiment value decrease with the increasing the length of parsed string. These reasons led us to use sentences instead of reviews. The sentences are then considered as individual reviews and thus final score for hotel is then evaluated.

D. Score methodology and normalisation

The output method used for sentiment pipeline was probabilistic. This method is more accurate than classifying sentences into five class (very negative, negative, neutral, positive, and very positive) which is the default behaviour of sentiment pipeline of Stanford's coreNLP. Following is a sample output for the string "the food was very good but the service was very bad."

```
(0 (1 (2 (3 (4 (5 the) (6 food)) (7 (8 was) (9 (10 very) (11 good)))) (12
but)) (13 (14 (15 the) (16 service)) (17 (18 was) (19 (20 very) (21
bad))))) (22 .))
  0:  0.1045  0.4864  0.3162  0.0748  0.0181
  1:  0.0821  0.5242  0.2640  0.1131  0.0165
  2:  0.0098  0.0652  0.2333  0.6296  0.0621
  3:  0.0092  0.0157  0.0551  0.7101  0.2100
  4:  0.0004  0.0037  0.9844  0.0111  0.0005
  5:  0.0004  0.0025  0.9941  0.0024  0.0006
  6:  0.0006  0.0032  0.9932  0.0016  0.0014
  7:  0.0095  0.0150  0.0294  0.7392  0.2070
  8:  0.0003  0.0022  0.9960  0.0009  0.0004
  9:  0.0043  0.0094  0.0161  0.7604  0.2097
  10:  0.0002  0.0018  0.9954  0.0022  0.0004
  11:  0.0007  0.0049  0.0020  0.9874  0.0050
  12:  0.0018  0.0052  0.9897  0.0016  0.0018
  13:  0.1540  0.6303  0.1915  0.0184  0.0058
  14:  0.0012  0.0077  0.9137  0.0742  0.0032
  15:  0.0004  0.0025  0.9941  0.0024  0.0006
  16:  0.0011  0.0029  0.9860  0.0056  0.0044
  17:  0.2212  0.6012  0.1641  0.0084  0.0052
  18:  0.0003  0.0022  0.9960  0.0009  0.0004
  19:  0.3896  0.3964  0.1911  0.0075  0.0154
  20:  0.0002  0.0018  0.9954  0.0022  0.0004
  21:  0.8057  0.1500  0.0070  0.0025  0.0348
  22:  0.0004  0.0012  0.9965  0.0013  0.0005
```

The first line of the output is the parsed string in the form of binary tree. The lines that follow it give the probabilistic score for the corresponding number of node.

The parsed tree string was read by pyparsing module of python and converted to nested string array.

```
# returns pyparsed nested string array for given string.
def getParseString(s):
    enclosed = Forward()
    nestedParens = nestedExpr('(', ')', content=enclosed)
    enclosed << (Word(alphanums+'.') | ',' | nestedParens)
    return enclosed.parseString(s).asList()
```

The score then is calculated for the root node (node number 0) by multiplying the probability of very negative, negative, neutral, positive, very positive sentiments with 1, 3.25, 5.5, 7.75 and 10 respectively and adding thus to get a total score for a line. The numbers (1, 3.25, 5.5, 7.75, and 10) are the equal distribution of the range 1…10 (the desired range of output).

The score achieved from this had a mean of 5.73 and median of 5.75. However, the standard deviation was just 0.49.

Mean	Median	Standard Deviation
5.73	5.75	0.49

Thus was the need for normalisation of score so as to fill the given range 1…10.

The probable cause of this small standard deviation is that many of the reviews are objective and do not portray any subjective review for the hotel. These objective reviews account for the said low standard deviation.

For normalisation of score we neglected neutral reviews in calculating the overall rating of the hotel but we are not normalising the ratings of different features as mostly sentences used to calculate those ratings are not objective for example in the review from holidayiq.com “I reached hotel at 2:00 pm, the location was very good but I was not satisfied by the food.”, the total rating includes the rating of the line “I reached the hotel at 2:00 pm”. So, we must exclude the neutral comments. After neglecting the neutral comments standard deviation increased with almost same mean and mode:

Mean	Median	Standard Deviation
6.08	6.30	0.91

E. Feature-aspect based sentiment analysis

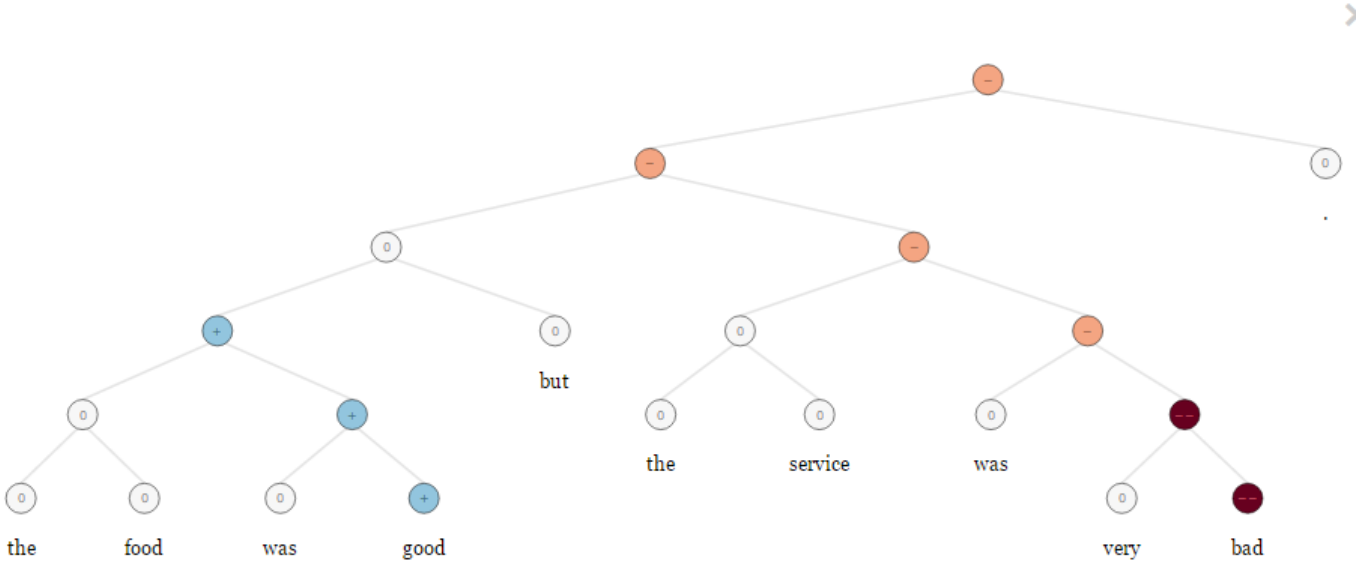
One of the key features of our solution is ranking and grading hotels based on certain features. Transforming of reviews into a single score led to discarding of vital information that the review sentence has. For instance, marking hotels that offer a good breakfast, or hotels that have friendly staff. This is a helpful way for users to find hotels which have the facilities that each individual user finds important. If one does not need a breakfast, but require good location, it should be possible to filter out what previous users think about specific services and filter results accordingly. These features are defined ahead of time and are generally meant to be domain specific. In our solution, we defined and identified opinions about the following features: food, location, service, room and cleanliness (the total rating is also present).

For this, specific words marking the feature of food, location, room, service and cleanliness was seeded in arrays. The occurrence of these words was then figured from the string. If the main string contained more than one feature, then the sentence was broken into smaller phrases recursively.

The initial idea for the feature extraction was to identify sentences containing each feature and determine scores based on these. Mostly this approach worked fine. However, some sentences contain multiple clauses, and not all parts are necessarily relevant. Sometimes they branch of into completely different topics, or were completely objective. This results in quite a bit of noise when it comes to calculating the individual feature's sentiment scores. Consider for instance the following sentence from a TripAdvisor review "the food was good but the service was very bad." This sentence contains multiple features. This approach would give average score to both food and service.

We then changed our approach to a finer level. The new approach to this was using nested array parsed from the tree string. The algorithm recursively traverse the tree until it finds the biggest chunk of phrase with single feature in it.

For instance, considering our previous example "the food was good but the service was very bad."



The parser detects the two fragments – “the food was good” and “the service was very bad”. Then it gives the score to individual feature thus specifying this particular hotel to be good in food while bad in service. The noteworthy point is that the total score given remains the same, i.e. the score of the root node (node numbered 0)

A suitable score for each sentence based on a sentiment analysis algorithm was given. The Stay ranking is the aggregate of each individual customer review. Similarly for different features, the score for each feature is calculated using the values extracted from sentiment binary tree and feature ranking is the aggregate of those scores.

IV. User Interface

The user interface is made while taking both non-technical and technical individuals in mind. For this both command line interface and web-based interface is created.

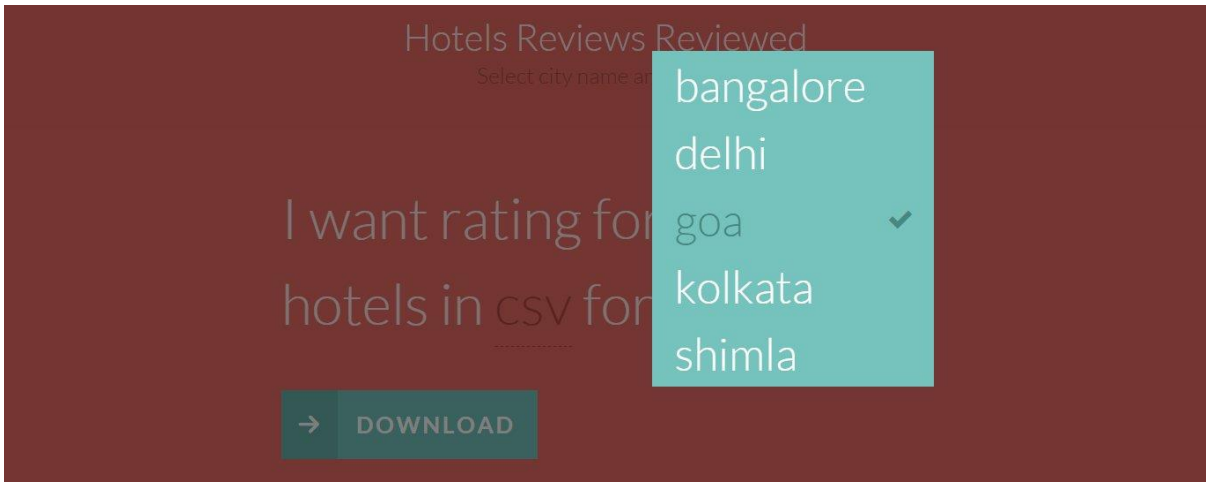
1. Command Line Interface (CLI)

Command line interface is python script which takes in city name as argument and then first fetches list of hotels, then the reviews and description for each of the hotel. Then the script runs the coreNLP package for sentiment analysis. The output is a combination of three files, created under a directory (named on the city-name). The CLI is currently only available for windows OS as of now. The complete documentation and source code of CLI is attached in annexure.

2. Web-based simple interface

This is made particularly for non-technical individuals. The design of website is kept simple. One can select few options to specify the city one needs, and the output format one desired (CSV or JSON). The files are updated on regular interval. Few screenshots of web-based simple interface are below. The output files from the web-based simple interface are attached in annexure.

The option buttons are ‘city name’ and ‘format’ that currently holds “goa” and “csv”.



3. Web-based advanced interface

It is quite clear that the above web-based interface is for very minimal amount of use. Any desire of complex query fails for it. Thus, was the need for creating a more complex web-based interface. The website reads the generated CSV and create HTML table from it. Then, additional jquery plugins for filter, sort, pagination and exporting the rendered view was used. Following is the screenshot-

export csv

title	location	region	stars	amenities	start price	end price	rating
		Andhe×	3.5				
Aura Grande	Plot No. 34/21, Central Road, MIDC, Andheri (East), 400 093, India	Andheri (East)	3.5	Room Service, Free Parking, Restaurant, Bar/Lounge	4195	7650	7.04897386
Mumbai Metro The Executive Hotel	Neelkanth Udyog Bhavan, Saki Vihar Road, Saki Naka, Andheri (east), 400072, India	Andheri (east)	3.5	Free Parking, Mid-range, Room Service, Free Parking, Free High Speed Internet (WiFi)	3393	5614	6.51667963
The Paradise by Tunga	P/16 MIDC Central Road, Andheri (East), 400093, India	Andheri (East)	3.5	Free Breakfast, Mid-range, Free Parking, Room Service, Restaurant, Free Parking, Bar/Lounge, Free Breakfast, Free High Speed Internet (WiFi)	3887	5614	6.44464935
Hotel Suncity Residency	16th Road, MIDC, Marol, Andheri East, 400093, India	Andheri East	3.5	Restaurant, Room Service, Free Parking, Bar/Lounge, Free Breakfast, Free High Speed Internet (WiFi)	4380	6169	6.32298653
Kohinoor Continental	Andheri Kurla Road, Andheri E, 400059, India	Andheri E	3.5	Vile Parle East, Free Breakfast, Mid-range, Pool, Pool, Restaurant, Room Service, Free Parking, Fitness Centre with Gym / Workout Room, Bar/Lounge, Free Breakfast	5244	11598	6.22948645
Hotel Oriental Aster	Near Domestic & International Airport,No 45 Tarun Bharat Co-operative Housing Society, Dr.Karanjiya Road, Chakala, Andheri (East), 400099, India	Andheri (East)	3.5	Free Breakfast, Mid-range, Free Parking, Room Service, Free Parking, Restaurant, Bar/Lounge, Free Breakfast, Free High Speed Internet (WiFi)	3948	4936	5.51132812

V. LOGIC

The target audience for the information extracted is three folds-

Common mass, trying to book tickets while choosing hotels based on filters and preferences.

Currently existing booking platforms, which could use our sentiment scores to add additional parameters on their platform.

Hotel Managers, which would like to know about reviews about their hotel.

For the traveling user AKA common mass, who is accessing reviews on the web for planning his travel, many of these considerations are not relevant, as he will be content with a momentary snapshot of reviews. But for hoteliers interested in user comments on the web a service that automatically and systematically collects reviews, classify them and rate accordingly would be advantageous and perhaps even more useful than the paper forms many hotels use for gathering feedback from their guests. One may not read all reviews given, but given the score- feature based and total, one can judicially arrive at decision.

It is evident that showing user's review increases the conversion rate by 14-76%. Now showing them handier information would be quite useful and necessary for booking platforms.

Though nearly every internet travel agency and hotel booking service nowadays offers also ratings and/or reviews of hotels, it is not that easy for hoteliers who want to know what is published about their hotels on the web to gather the user generated information. A standard search engine like Google will give thousands of hits for a hotel. But, though there seems to be a huge number of sites providing user reviews, often these are just the same because many sites use the same source. In other cases, the links lead only to some general page from which one can access reviews besides other information and lacking transparent navigation structure. Also, the links might point to some individual review but leaving it open whether there are other reviews on the site.

Another problem concerns the kind of information: travel agencies and hotel booking services often only publish scalar ratings, e.g. scores between 1 and 5. Such scores are not very helpful for hotel managers as the numeric value does not provide information of what guests actually considered positive or objectionable. For hotel managers the textual user comments would be much more significant than the numeric scores since they would be interested to know what the users exactly commented on and how they thought of it. Another

problem for hotel managers is that of following updates and new reviews. Hotel booking services and travel agencies collect and publish user reviews systematically, e.g. by asking their customers for comments or ratings. So, new reviews appear quite frequently on their pages but it would be difficult to follow these by just using general search.

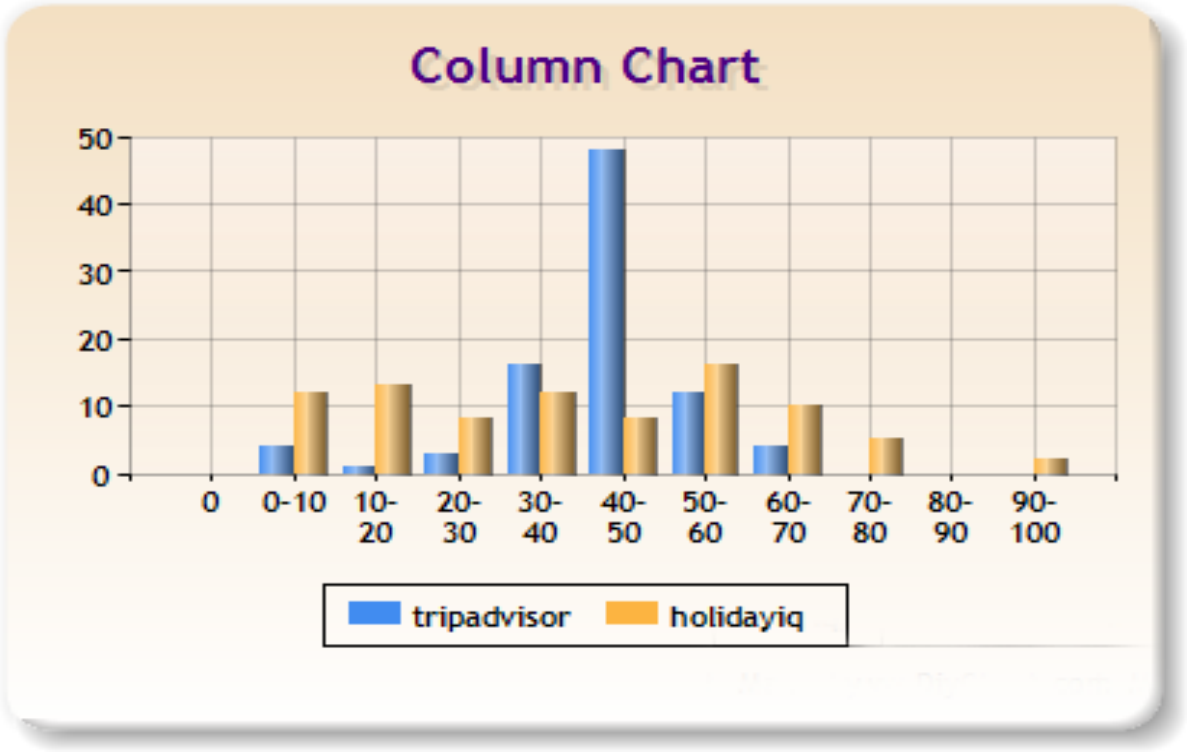
So, we aim at providing such a service for hotel managers that collects user reviews for hotels from various sites on the web, analyse the textual content of the review and give classified ratings along with classified reviews.

VI. EVALUATION

Given the bulk of data, we then tried various evaluation tools so as to verify our findings as well as achieve to a conclusion with them.

We evaluated the analysis system on a corpus of 86 hotels (North Mumbai) reviews crawled from the web. These reviews contained 7112 text segments. For the evaluation, these segments were manually classified with respect to their polarity, including the neutral polarity besides positive and negative ones. Also, we annotated the segments whether they cover more than one topic. The distribution from this manual classification is shown in Table. Evaluated on all segments, the results in following Table were achieved.

Website	Total reviews	Positive reviews	Negative reviews	Neutral reviews
holidayiq	3334	1583	698	1053
tripadvisor	3788	1171	1195	1422



We are providing full classification of the reviews to users. Take an example of two hotels both from Andheri East, North Mumbai, we provided feature based ratings along with number of very positive, positive, very negative, negative and neutral reviews for each feature.

Title	Star rating	Rating	Total number of reviews	Positive reviews	Negative reviews	Location rating	Total food Reviews	Food negative reviews	Food neutral reviews
The Paradise by Tunga	3.5	6.44	64	34	30	6.303495	5	4	1
Hotel Cosmo	2.5	4.75	20	9	11	4.238138	2	1	1

In case of our paid users i.e. hotel managers we are providing classified reviews along with ratings as they are interested in user comments.

One of the best uses of feature based ratings (as explained above) is that it is a helpful way for users to find hotels which have the facilities that each individual user finds important. Let's take an example of two hotels in North Mumbai, one hotel 7 Flags International Hotel having overall rating 5.7 and location rating 4.4 and other is Hotel Adore Inn with overall rating 5.5 and location rating 8.7, So if a person gives more importance to location then he would chose second option.

We also verified the working of feature based Stanford sentiment analysis tool manually. We manually classified 110 sentences with respect to their features and then further classified them into positive and negative sentences. The following table shows the results.

	Total number of sentences	Positive sentences	Negative sentences	Positive sentences related to food	Negative sentences related to food	Positive sentences related to location	Negative sentences related to location
Feature based Stanford sentiment analysis	110	42	31	8	10	9	10
Manually done	110	36	34	9	13	12	8
Difference		6	3	1	3	3	2

VII. CONCLUSION

We presented a web based opinion mining system for hotel reviews and user comments that supports the hotel management in monitoring what is published on the web about their hotels. The system is capable of detecting and retrieving reviews on the web, to classify and analyse them and to give ratings accordingly. The system provides good performance for the analysis and the classification tasks.

Future Plans

We, given the limited processing, memory and time resources could only limit our finding to few cities, few hotels and few reviews out of a big pool of data. So, our immediate plan would be to scale for more data.

To get more accurate results, we will train the sentiment tree bank with specific hotel reviews.

We have used social networking website (twitter.com) to fetch data but we have not incorporated them for sentiment value analysis, because analysis of social networking posts/tweets is a lot different from normal reviews. With social networking in hand, we would add our solution to read smiiles/emoticons, images and different form of texts present in social networking website.

For hotel managers, we would be adding visualised results with charts and trend-lines displaying how the hotel is doing with time.

ANNEXURE

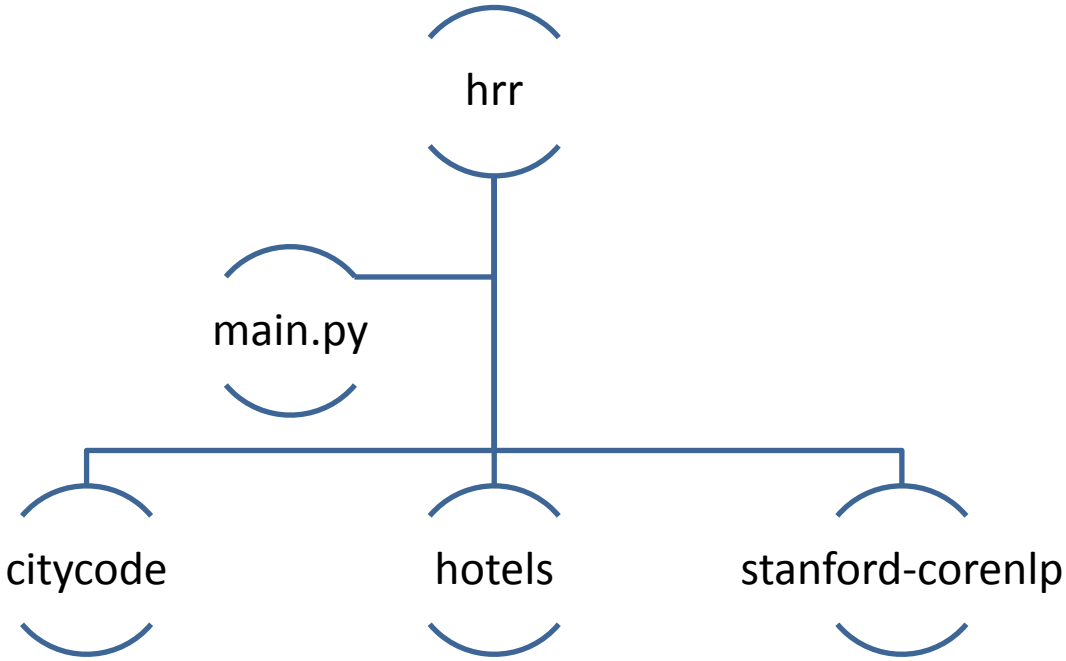
1. **Documentation for CLI**

Download the ‘hrr’ package and unzip it into a directory.

For the command line interface, following are the dependencies

- Python 2.7, https://www.python.org/downloads/
- Pip, python get-pip.py
- Scrapy, pip install Scrapy
- Pyparsing, http://pyparsing.wikispaces.com/
- Stanford coreNLP Package, http://nlp.stanford.edu:8080/corenlp/ (in ‘hrr’ directory)

The hierarchy of the



The usage of ‘hrr’ package is just a single line.

The python command for it is-

- python main.py $cityname
  where, $cityname is the variable for which we have to fetch hotels and do sentiment analysis.
  N.B. $cityname must be capitalised and given as single word with hyphens.

2. **Attachments**

- **hrr.zip** : for command line interface
- **mumbai.zip:** Sample data file, comprising of review.json, rate.csv for *North Mumbai*
    - **review.json:** JSON array of all the reviews and description extracted.
    - **Rate.csv**: A CSV file that has the rating of different hotels, categorised by features.
- **link.py**: Scrapy code for data fetching
- **sentiment.py**: Python code for sentiment analysis on reviews
- **screenshots.zip**: Zip file of screenshots of website.
- **twitter.zip**: For evaluation of twitter sentiments

References

1. CoreNLP, http://nlp.stanford.edu:8080/corenlp/
2. Scrapy, http://scrapy.org/
3. Python Documentation, https://docs.python.org/2/
4. Sentiment Analysis for Hotel Reviews, Walter Kasper, Mihaela Vela, http://www.dfki.de/web/research/publications/renameFileForDownload?filename=25.pdf&file_id=uploads_1206
5. Recursive Deep Models for Semantic Compositionality, Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng and Christopher Potts, http://nlp.stanford.edu/~socherr/EMNLP2013_RNTN.pdf
6. Star Quality: Aggregating Reviews to Rank Products and Merchants, http://www.aaai.org/ocs/index.php/ICWSM/ICWSM10/paper/viewFile/1507/1844
7. A Study of Opinion Mining and Visualization of Hotel Reviews, Eivind Bjørkelund, Thomas H. Burnett, Kjetil Nørvåg, https://www.idi.ntnu.no/~noervaag/papers/iiWAS2012.pdf
8. SENTIMENT ANALYSIS: SARCASM DETECTION OF TWEETS, http://www.academia.edu/7499991/SENTIMENT_ANALYSIS_SARCASM_DETECTION_OF_TWEETS
9. expedia, http://www.expedia.co.in/
10. tripadvisor, http://www.tripadvisor.in/
11. holidayiq, http://www.holidayiq.com/
12. tablesorter, http://tablesorter.com/docs/
13. datatables, http://datatables.net/examples/index