Aspect based Sentiment Analysis and Summarization of Reviews

Stanford CS221 Project - Ankit Dwivedi, Arunothia Marappan

December 14, 2019

Motivation

User reviews are now an integral part of how services like Restaurants, Hotels, Online Shopping, Taxi, Airbnb, Google Maps and a lot more operate. It has become a reflex action to many of us to check the rating/reviews of a restaurant before visiting it. This is essentially because users are willing to be more honest and detailed in writing their reviews and retailers are willing to trust and work upon the feedback they get like never before.

In this project, we explore ways to make more sense of Restaurant reviews. We do this by detecting the sentiment of the reviews w.r.t different aspects of running a Restaurant service like Ambience, food, Price, Service, etc. We also make an attempt to extract the summary of multiple reviews to make it easier for users to get an idea of the Restaurant they would like to visit.

Task Definition

Input-Output Behaviour

Problem1: Aspect based Sentiment Analysis

-

Input

Review text of any Restaurant.

Output

Sentiment **(Positive, Negative, Neutral)** of the review on each of the aspects - **(Ambience, Anecdotes/Miscellaneous, Food, Price, Service)**

Problem2: Summarization

- **Input**: A set of review texts of any given hotel in the format of our dataset.
- **Output**: A summarization of the review texts.

Evaluation Metric for Success

Subtask 1: Aspect Detection

We use F1 score to report a combined metric for the precision and recall of our aspect detection models

Subtask 2: Sentiment Analysis

Success is measured by test accuracy or the number of sentiments that our model gets right.

Summarization

Due to lack of labelled data, we will not report any metric for this. We will provide an example of a summary generated by our baseline model.

Approach - Developing Intuition

To build a better intuition about the problem in hand, we implemented a baseline and oracle models -

Baseline

Problem1: Aspect based Sentiment Analysis

We manually labelled the aspect specific sentiments of 50 reviews from kaggle/515k dataset. For getting a baseline, we used `Scikit-Learn MultinomialNB` to run a bag of words method with (unigrams, bigrams, trigrams) as feature set. With this simple baseline, we got a Test Accuracy : 57.5%

Problem2: Summarization

We used a simple algorithm that uses GloVe word embeddings to find the vector representation of each sentence. It then calculates the similarities between sentence vectors using cosine similarity, and stores them in a matrix, which is converted into a graph, with sentences as vertices and similarity scores as edges, for sentence rank calculation using TextRank [8]. Finally, it picks a certain number of top-ranked sentences to form the summary.

Baseline Sample Input

30 reviews of a given hotel (a lot of reviews talk about breakfast)

Baseline Sample Output

The continental breakfast was great, but they should consider expanding their hours or at least allowing guests to finish the food that is out at 10, because every morning the food was taken away promptly at 10:00 even though the area was packed with guests still trying to get breakfast. The hotel offers free breakfast that was really good (not the usual crappy free breakfast) The only thing I didn't like about the room was that the air conditioner was regulated a motion sensor.

Oracle

Problem1: Aspect based Sentiment Analysis

For getting an oracle estimate, we cheat and train on the test data. We do the training using the same bag of words model as mentioned in our baseline. And by this we get a 100% accurate oracle.

Problem2: Summarization

Human performance (human generated summaries) is the oracle for the summarization task.

Infrastructure

We implemented our baseline using kaggle/515k dataset. To use this dataset for our problem statement we had to manually label the sentiment of each review with respect to the aspects we had chosen. This clearly was not scalable beyond baseline and hence, we moved to Semeval-2014 dataset that has aspect specific sentiments labelled for 3041 reviews. To help our models we later also added Semeval-2015 and Semeval-2016 datasets to increase the number of reviews to 4769 (The datasets had many overlapping reviews which we had to remove). Additionally, we also use kaggle.com/yelp-dataset that is sentiment labelled to train some of our n-gram based features.

Data

kaggle.com/yelp-dataset

- `Preprocessing.yelp_restaurant_review_parser.py` parses yelp data.We tried this data for sentiment based PMI scores but since this did not improve the accuracy beyond what we get with our original data, we did not pursue it further.

Semeval-2014, Semeval-2015 and Semeval-2016

- `Preprocessing.processDataFromSemEval.py` parses Semeval-2014 data.
- `Preprocessing.processDataFromSemEval2015.py` parses Semeval-2015 and Semeval-2016 data.
  - The aspect names are different in this when compared to 2014 data so we mapped them to match.
  - Also, 2015 and 2016 datasets did not have the sentiment "Conflict" and hence, we changed our input-output behavior to only include sentiments $\{Positive, Negative, Neutral\}$.
- `Preprocessing.getFilesForPMI.py` parses input data to estimate sentiment based PMI scores.
- `Preprocessing.getFilesForAspectBasedPMI.py` parses input data to estimate aspect based PMI scores.

Preprocess Data

- `util.preprocessText(text)` removes special characters, extra spaces, converts `text` to lower case.
- `util.lemmatize(text)` lemmatizes every token in `text` using `WordNetLemmatizer` from `nltk` library.
- We also tried `autocorrect.Speller` but that slowed down our preprocessing and was not very useful.

Literature Review

Aspect based Sentiment Analysis

Most of the early research in this space was focused on using variations of logistic classifiers and Support Vector Machines (SVM) trained on ngram features. In [11], it is shown that inclusion of word bigram features gives consistent gains on sentiment analysis and text classification tasks, and a simple SVM variant using Naive Bayes (NB) log-count ratios as feature values performs equivalent to the contemporary state-of-the-art models.

In [6], the authors use Pointwise Mutual Information (PMI) scores as features to represent word-aspect and word-context association, and train linear SVM models for aspect detection and sentiment analysis.

Modern research has found a host of deep neural network architectures that outperform classical methods. For instance, [10] presents the results from a long short-term memory (LSTM) network and an extension that incorporates semantic relatedness of an aspect with its context words.

Summarization

Text Summarization is a hot research topic in natural language processing. There are 2 approaches to summarization:

1. **Extractive Summarization**: The extractive text summarization technique involves pulling key phrases and important sentences from the source document and combining them to make a summary. Most of the algorithms in this space involve using word vectors to identify important phrases and sentences.

2. **Abstractive Summarization**: The abstraction technique entails paraphrasing and shortening parts of the source document. When abstraction is applied for text summarization using deep neural networks, it can overcome the grammar inconsistencies of the extractive method, which makes the human summart much more readable.

   Recently, Microsoft Research's UniLM AI model achieved state-of-the art performance on text summarization and language modelling.

Approach

Aspect Detection

Aspect detection is the classification task of detecting if a given review talks about a particular aspect. For instance, *food was good, but service was horrible* is talking about the **food** and **service** aspects.
For this subtask, we tried multiple approaches which are briefly explained below.

1. **Multinomial Naive Bayes Classifier**
   A multinomial naive bayes classifier (MNB) [5] is a basic yet effective technique for document classification that relies on a bag of words representation of the document. Given C classes, and n features $x_1, x_2, \ldots x_n$ (which represent frequencies of words or ngrams in the document), the output of MNB is given by

   $$_{c\in C}P(x_1, x_2 \ldots, x_n|c)P(c)$$

   where $P(x_i|c_j)$ are assumed to be conditionally independent given the class c_j.

   The bag of words model in our implementation consists of unigrams, bigrams and trigrams.

2. **Linear Support Vector Machine**
   The objective of the support vector machine (SVM) [4] algorithm is to find the hyperplane that has the maximum margin in an N-dimensional space that distinctly classifies the data points. We experimented with different features to train the SVM:

   (a) **Term frequency, Inverse Document Frequency (tf–idf)**: Tf-idf is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

   - tf(t, d) - is a measure of how frequent n-gram t occurs in document d. If $f_{t,d}$ denotes the raw count of n-gram t in document d, then

     $$tf(t,d) = f_{t,d}$$

   - idf(t, D) - is a measure of how frequent or rare n-gram t is across the corpus of N documents D.

     $$idf(t,D) = log\frac{N}{\{d \in D : t \in d\}}$$

$$\text{tfidf}(t,d,D) = \text{tf}(t,d) \cdot \text{idf}(t,D)$$

For example, consider the following simple Document set - $D = \{d_0$:”food is good”, d_1:”bad food”$\}$.
Let us run through our $(tf - idf)$ scores on this.

- $tf(food, d_0) = 1$ as food occurs once in document d_0.
- $idf(food, D) = log(\frac{|D|}{\{d \in D: t \in d\}}) = log(\frac{2}{2}) = 0$.
  0 value is indicative that $food$ is not a unique word and that it is present in every document.
- $tf - idf(food, 0, D) = tf(food, 0) * idf(food, D) = 0$
- $idf(bad, D) = log(\frac{|D|}{\{d \in D: t \in d\}}) = log(\frac{2}{1}) = 1$
  Non zero value indicating that *bad* is relatively rare in the corpus.

(b) **Word-Aspect Pointwise Mutual Information (PMI) [6]**:

- $freq(w,a)$ is defined as the frequency of the n-gram **w** occurring in all reviews having $aspect = a$
- $freq(w)$ is defined as the frequency of the n-gram **w** occurring in all reviews.
- $freq(a)$ is the count of all n-grams **w** occurring in all reviews having $aspect = a$.
- N is the total n-grams in all reviews.

$$score(\mathbf{w}) = PMI(\mathbf{w}, aspect) - PMI(\mathbf{w}, \neg aspect)$$

$$PMI(\mathbf{w}, aspect) = Log2(\frac{freq(w, aspect) * N}{freq(w) * freq(aspect)})$$

(c) **Log-Count Ratio [11]**:

- $f^{(i)}(w)$ is the count frequency of w in document i.
- $y^{(i)}$ is the sentiment of document i.
- α is the smoothing parameter.

$$p = \alpha + \Sigma_{i:y^{(i)}=1} f^{(i)}$$
$$q = \alpha + \Sigma_{i:y^{(i)}=-1} f^{(i)}$$
$$log\text{-}count\ Ratio = \log\left(\frac{p/||p||_1}{q/||q||_1}\right)$$

3. **Neural Networks**

[2] gives a basic overview of neural networks. They consist of layers that are made up of interconnected nodes which map their input to an intermediate representation using a non-linear activation function, which becomes the input for the next layer. The final output is thus a non-linear combination of the input feature vector and weight vectors at each layer.

Recurrent Neural Networks (RNN) are a special class of neural networks that are used for sequential data. Since natural language is sequential, RNNs are well suited for almost all natural lanugage processing tasks.

RNNs perform the same computation for every element of a sequence, with the output depended on previous computations [7]. Figure 1 shows the basic RNN structure. At a particular time step t, X_t is the input, y_t is the output, and a_t is the activation of the RNN cell, which contains neural networks just like a feed-forward network.

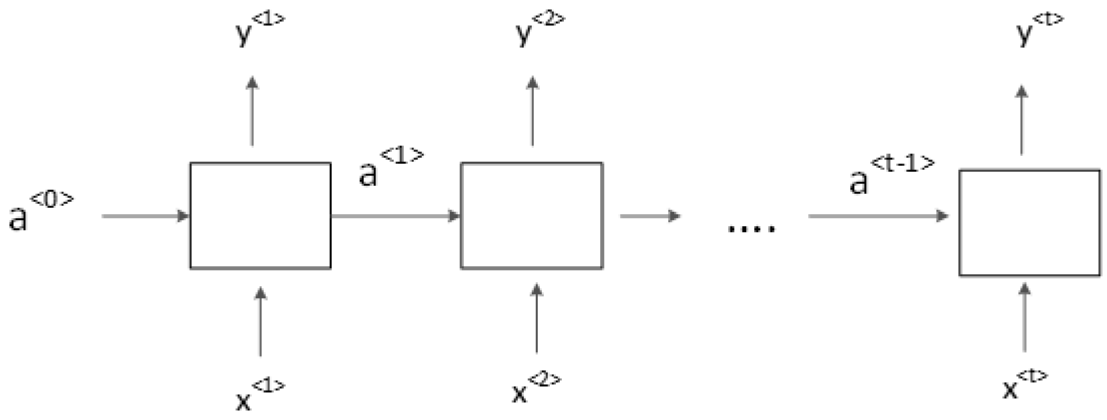


Figure 1: RNN structure

For each timestep t the activation $a^{<t>}$ and the output $y^{<t>}$ are expressed as:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a$$
$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

where W_{ax}, W_{aa}, W_{ya}, b_a, b_y are coefficients that are shared temporally and g_1, g_2 are activation functions. [1]

Variations of RNN
Simple RNNs suffer from vanishing gradient problem for long sequences. As the sequence length increases, the network becomes deeper, and the gradients flowing back in the back propagation step become smaller. Effectively, a basic RNN has many local influences. That is, output of each layer is mainly affected by immediately earlier layers, and so the output is also based on the most recent words.

Following variations of the simple RNN cell address the problem of vanishing gradients:

(a) **Gated Recurrent Unit (GRU)**: The GRU cells take as input the previous state and the current input. Internally, these cells decide what to keep in and what to eliminate from the memory using 2 gates:
   - Update gate (Γ_u): Decides if the previous cell's activation should be kept in memory
   - Relevance gate (Γ_r): Decides how relevant is the previous cell's activation in calculating this cell's activation.

(b) **Long Short Term Memory (LSTM)**: LSTM [3] is a generalization of GRU. It tries to achieve the same purpose as a GRU, but uses 3 gates instead of 2 - update gate (Γ_u), forget gate (Γ_f), and output gate (Γ_o).

Input Representation using Word Embeddings
Word embeddings are vector representations of words in the input vocabulary, such that semantically similar words are represented using similar vectors. We used 100 dimensional GloVe word vectors (6B tokens, 400K words) as the input to our neural networks. [9]

Neural Network architecture for Aspect Detection
We experimented with different variations of RNN (SimpleRNN, GRU, LSTM, Bidirectional GRU, Bidirectional LSTM). A GRU based network gave the best F1 score on the aspect detection task. Figure 2 shows our network architecture. In addition to a GRU layer, our network consists of the following layers:

(a) Embedding Layer

(b) Max and Average Pooling Layers

(c) Dropout Layer

(d) Fully Connected Layer

(e) Softmax Layer

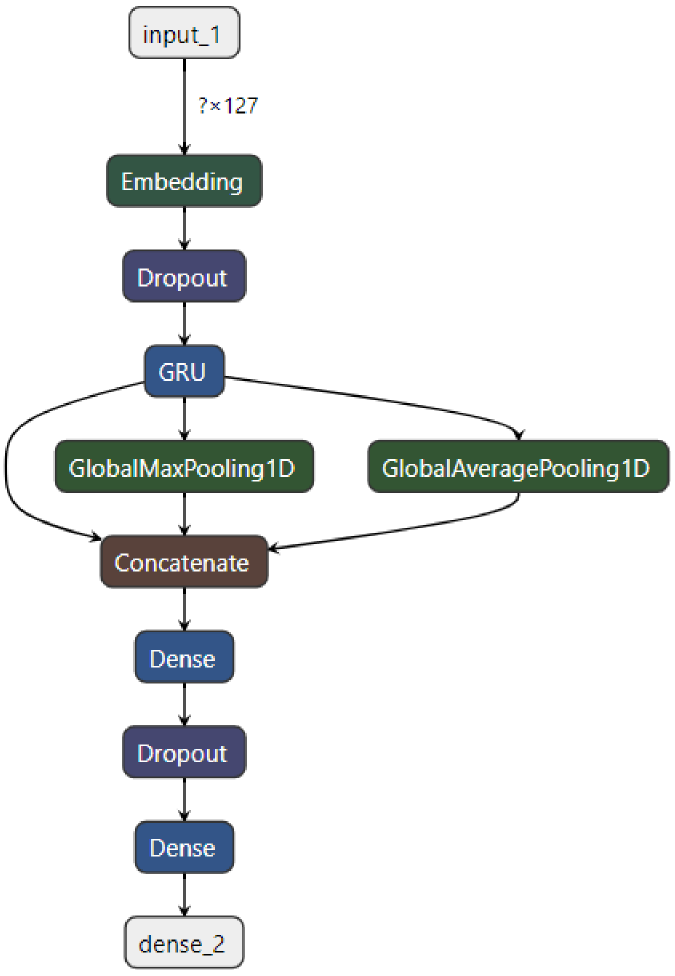


Figure 2: Network architecture

Sentiment Analysis

We tried similar models to aspect detection with the following additional sentiment specific feature extractors -

1. **TF-IDF using Negated Context [6]**

   - Attempts to capture whether a word is occurring in a negated context or not.
   - Add new n-grams with w_{NEG} for words w occurring between a negation word and a punctuation mark.

   $$\textit{Not good as expected.} \rightarrow \textit{Not } good_{NEG}\ as_{NEG}\ expected_{NEG}.$$

   - Find *tf-idf* scores on this new vocabulary.

2. **Sentiment Based Pointwise Mutual Information (PMI)** [6]

   - $freq(w,s)$ is defined as the frequency of the n-gram $\mathbf{w}$ occurring in all reviews having $sentiment = s$
   - $freq(s)$ is the count of all n-grams $\mathbf{w}$ occurring in all reviews having $sentiment = s$.

$$score(\mathbf{w}) = PMI(\mathbf{w}, sentiment) - PMI(\mathbf{w}, \neg sentiment)$$

$$PMI(\mathbf{w}, sentiment) = Log2(\frac{freq(w, sentiment) * N}{freq(w) * freq(sentiment)})$$

Results

Please look at Appendix for detailed figures (3, 4 5 6) and tables (1, 2) of our results.

Best Aspect Detection Results by **GRU** with Test F1 Score of 84.27%

Best Sentiment Analysis Results by both **Simple RNN** and **SVM with TF-IDF** with an overall Test Accuracy of 87.67%

Error Analysis

Word Features improve with more N-grams

In our analysis we tried to understand how word features perform when more n-grams are added to obtain them. We ran experiments on **TF-IDF** feature extractor and figured that including $1,2$ and 3-grams performed slightly better than including only 1 grams. We will use the following sample review to exemplify this.

TF-IDF with 1-3 grams returns Negative while with only 1 grams returns Negative

I asked for a menu and the same waitress looked at my like I was insane.

Let us analyse this example in a detail by looking at the weights learnt by the SVM.

1-gram Only

TF-IDF Feature Names : (”asked”, ”insane”, ”like”, ”looked”, ”menu”, ”waitress”)

Each of these get an equal Tf-IDF value of 0.408

Negative vs Positive + Neutral

- Weights: $(0.0308, 0.9704, 1.1780, 0.4215, -1.1586, 0.8808)$
- Score: 0.9477
- Winner: Positive + Neutral

Positive vs Negative + Neutral

- Weights: $(-1.7502, -0.6184, -0.8237, 0.2121, 1.4122, -0.4257)$
- Score: -0.813
- Winner: Positive

Neutral vs Negative + Positive

- Weights: $(0.7916, 0, -1.3374, -0.1024, -0.3908, -0.0817)$
- Score: -0.457

- Winner: Neutral

Overall Winner: Positive

1,2,3 grams

TF-IDF Feature Names : ("asked", "insane", "like", "looked", "menu", "waitress", "asked menu", "asked menu waitress", "like insane", "looked like", "looked like insane", "menu waitress", "waitress looked", "waitress looked like")

Each of these get an equal Tf-IDF value of 0.2581

Negative vs Positive + Neutral

- Weights: $(-1.3719, -0.2312, -0.2312, -0.0671, -0.2312, , -0.2798,$
- $-0.3289, 0.6815, -0.2312, -0.2312, -0.3381, -0.2312, -0.2312)$
- Score: -0.9174
- Winner: Negative

Positive vs Negative + Neutral

- Weights: $(1.1116, 0.1211, 0.1211, 0.6825, 0.1211, 0.3646, 0.2234,$
  $0.1211, 0.5201, 0.1211, 0.1211, 0.5004, 0.1211, 0.1211)$
- Score: 0.8597
- Winner: Negative + Neutral

Neutral vs Negative + Positive

- Weights: $(0.273, 0, 0, 0, -0.4731, 0, -0.0653, 0 - 0.0653, 0, 0, 0.0844, 0, 0)$
- Score: 0.0627
- Winner: Negative + Positive

Overall Winner: Negative

We can see that adding additional features that are clearly negative - like "looked like insane", "waitress looked like", etc helped the model to classify the review correctly as negative. Also, the Tf-idf score gets lowered with increased number of non-zero entries, making every *n*-gram less powerful to sway the model into one direction by itself.

PMI does not add value without Enhanced Preprocessing

In our experiments we noticed that PMI does not add the value we expected it would. One example review where PMI fails when compared to TF-IDF is -

TF-IDF with 1-3 grams returns Positive while PMI returns Negative

Largest and freshest pieces of sushi, and delicious!"

Intuitive Explanation to this Behaviour One of the main reasons why we are unable to get the full benefits of PMI is that our preprocessing is not very good. Even though we used a lemmatizer for our train data, it is not very good. Also, our implementation does not lemmatize the PMI specific data which makes words like "freshest" to not get tagged with "fresh". This makes a lot of words unseen in the context of sentiment and hence, PMI fails to provide the relevant sentiment specific insight.

GRU works for long reviews, SimpleRNN does not

The following table compares the precision, recall, and F1 scores of the SimpleRNN model and GRU model for aspect detection:

Model	Precision	Recall	F1 score
SimpleRNN	80.9	80.76	80.83
GRU	82.54	86.08	84.27

If we look at the number of reviews in the test set which talked about food aspect, and compare the results of these 2 models, it is clear that the update and relevance gates of a GRU cell help it keep earlier words in memory for long enough to make the correct prediction.

	Count	Average Review Length
Reviews in test set	477	13.51
Reviews with food aspect	200	14.36
Food aspect identified by SimpleRNN, but missed by GRU	1	7
Food aspect identified by GRU, but missed by SimpleRNN	15	25

Below is an example of 57 word long review where the GRU model could detect the food aspect, but SimpleRNN could not:

> My main concern was the sanity of the **food** that was being sent out to myself and others, but I would be lying is I said that as someone who has worked in restaurants since the age of fifteen I was expecting at least a minimal effort on the part of the restaurant to amend the situation.

Reviews with misc aspect are hard to classify

Reviews like "The all you can eat deal is truly amazing here", "A great place to meet up for some food and drinks" have food related words, but food is not the primary subject of the review. These kind of reviews with miscellaneous aspects are being mislabelled even by the more advanced models like GRU.

Conclusion

Working on this project helped us explore and understand various machine learning techniques in the context of binary and multi-class classification problems.

1. While basic models like Naive Bayes, linear SVM etc. perform well on the aspect detection task, we saw considerable gains by using RNNs.
2. While using the output of the best aspect detection model in the sentiment detection task, we observed similar accuracy given by basic and advanced models. This highlights the need for employing more novel approaches (models, features and network architecture) for better results.

3. The limited train and test set size proved to be a limitation. Without a validation set, we used library provided default hyperparameter values for all models.

Due to the lack of time, we could not explore summarization beyond our baseline extractive summarization model.

Resources

Code - https://github.com/ankitdwivedi23/cs221-project.

Model Experiments - https://worksheets.codalab.org/worksheets/0xd5f59e034cb5491fbc6972a334cc8691

Feature Experiments - https://worksheets.codalab.org/worksheets/0xd05a6069b1354bfe811247fc718604b3

Error Analysis - https://worksheets.codalab.org/worksheets/0x94844e9bdd4543dda7b877c6cb7ee5e7

Data - https://drive.google.com/open?id=1ZjXXPeT7EgCLYRTTYy$_{R}Mxyw6JFlh_{x}$8

References

[1] Amidi, S., Amidi, A.: Recurrent neural networks, https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks#

[2] Goltsman, K.: Introduction to artificial neural networks, https://datascience.foundation/downloadpdf/9/whitepaper

[3] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**, 1735–80 (12 1997). https://doi.org/10.1162/neco.1997.9.8.1735

[4] Hsu, C.W., Chang, C.C., Lin, C.J.: A practical guide to support vector classification https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[5] Jurafsky, D.: Text classification and naïve bayes, https://web.stanford.edu/class/cs124/lec/naivebayes.pdf

[6] Kiritchenko, S., Zhu, X., Cherry, C., Mohammad, S.: NRC-canada-2014: Detecting aspects and sentiment in customer reviews. In: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014). pp. 437–442. Association for Computational Linguistics, Dublin, Ireland (Aug 2014). https://doi.org/10.3115/v1/S14-2076, https://www.aclweb.org/anthology/S14-2076

[7] Le, J.: Recurrent neural networks: The powerhouse of language modeling, https://builtin.com/data-science/recurrent-neural-networks-powerhouse-language-modeling

[8] Mihalcea, R., Tarau, P.: Textrank: Bringing order into texts, https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf

[9] Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543. Association for Computational Linguistics, Doha, Qatar (Oct 2014). https://doi.org/10.3115/v1/D14-1162, https://www.aclweb.org/anthology/D14-1162

[10] Tang, D., Qin, B., Feng, X., Liu, T.: Target-dependent sentiment classification with long short term memory. CoRR **abs/1512.01100** (2015), http://arxiv.org/abs/1512.01100

[11] Wang, S., Manning, C.: Baselines and bigrams: Simple, good sentiment and topic classification. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 90–94. Association for Computational Linguistics, Jeju Island, Korea (Jul 2012), https://www.aclweb.org/anthology/P12-2018

Appendix - Results

Aspect Detection		
Models	Train F1 Score	Test F1 Score
Multinomial NB: BoW	96.08%	76.28%
SVM: Aspect Based PMI	86.10%	68.13%
SVM: Log Count Ratio	84.87%	69.98%
SVM: BoW	98.08%	79.25%
SVM: TF-IDF	97.70%	80.07%
Simple RNN	94.19%	80.83%
LSTM	98.29%	83.38%
Bi-LSTM	98.32%	83.87%
GRU	98.26%	84.27%
Bi-GRU	98.16%	81.60%

Table 1: Aspect Detection Results

Sentiment Analysis

Sentiment Analysis Per Aspect						
Models	Ambience	Misc	Food	Price	Service	All
Multinomial NB: BoW	91.82%	72.75%	85.74%	94.97%	89.10%	86.88 %
SVM: Sentiment Based PMI	91%	72%	81%	95%	86%	85.03%
SVM: Log Count Ratio	92%	73%	82%	95%	88%	85.91%
SVM: BoW	92.66%	75.68%	85.53%	95.18 %	88.47%	87.51%
SVM: TF-IDF	92.54%	75.68%	86.37%	95.18%	88.68%	87.67%
SVM: TF-IDF (Negated Context)	92.24%	76.31%	85.53%	95.39%	88.68%	87.63%
Simple RNN	91.82%	76.52%	84.91%	94.76%	90.36%	87.67%
GRU	91.82%	77.15%	84.70%	95.18%	89.31%	87.63%

Table 2: Sentiment Analysis Results

Appendix - Figures

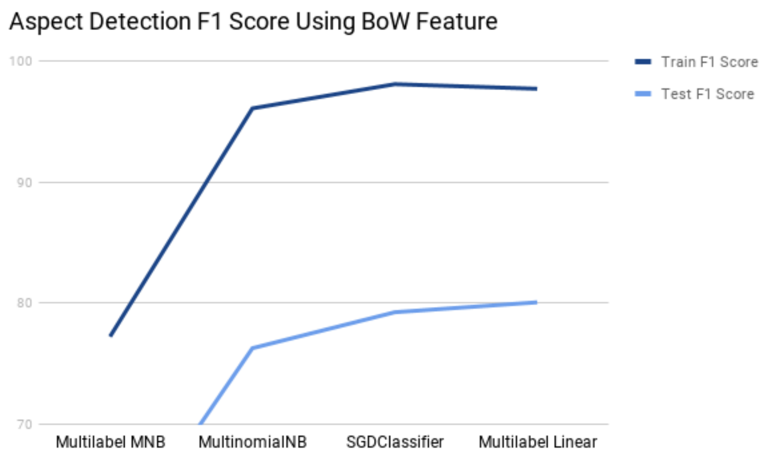


Figure 3: Aspect Detection Using Bag of Words

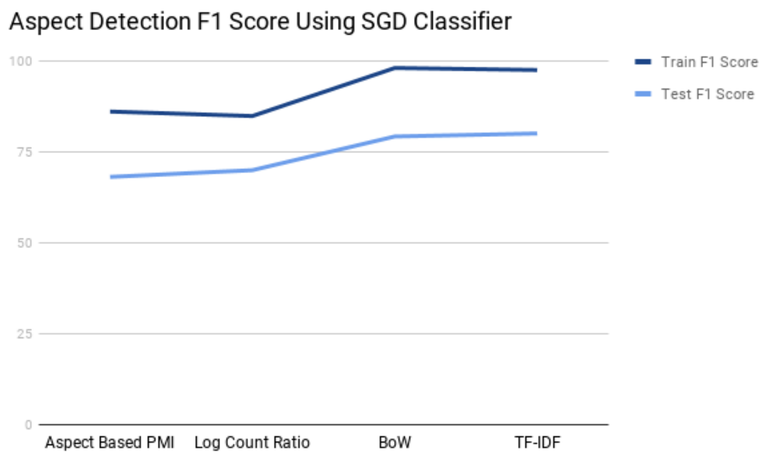


Figure 4: Aspect Detection Using SGD Classifier

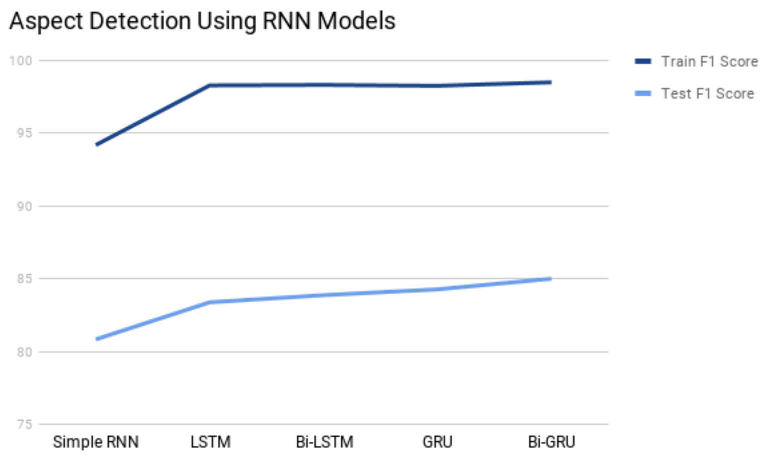


Figure 5: Aspect Detection Using RNN Models

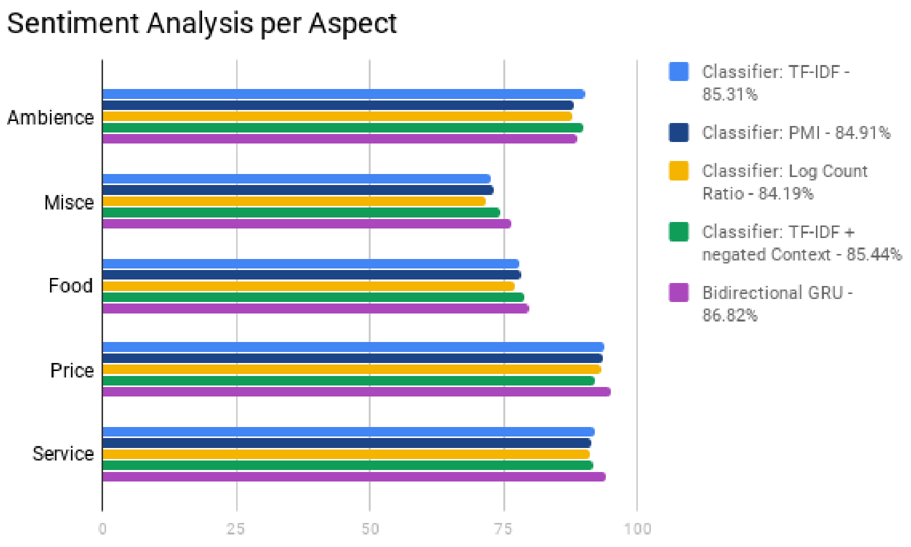
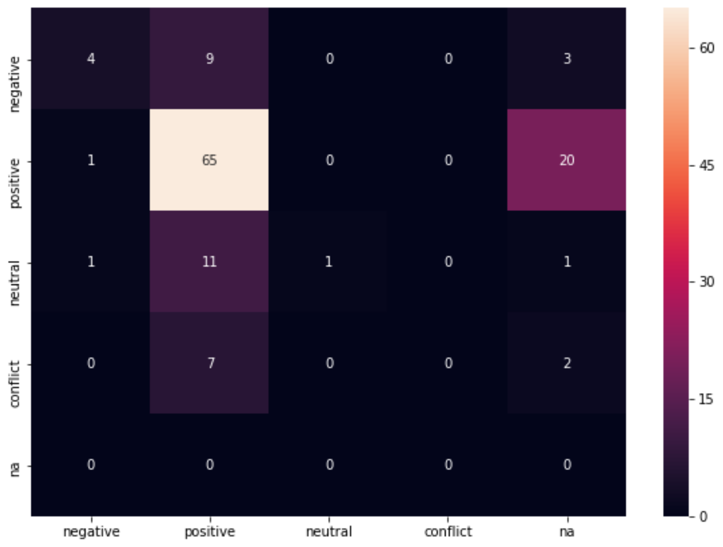


Figure 6: Sentiment Analysis Using Different Models/Features

Sentiment Analysis Using TF-IDF Feature



	negative	positive	neutral	conflict	na
negative	4	9	0	0	3
positive	1	65	0	0	20
neutral	1	11	1	0	1
conflict	0	7	0	0	2
na	0	0	0	0	0

Figure 7: Confusion Matrix: Food

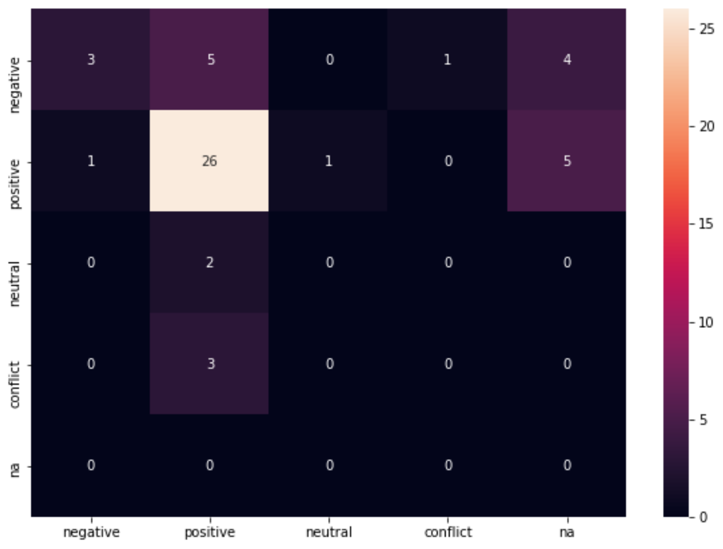


Figure 8: Confusion Matrix: Service

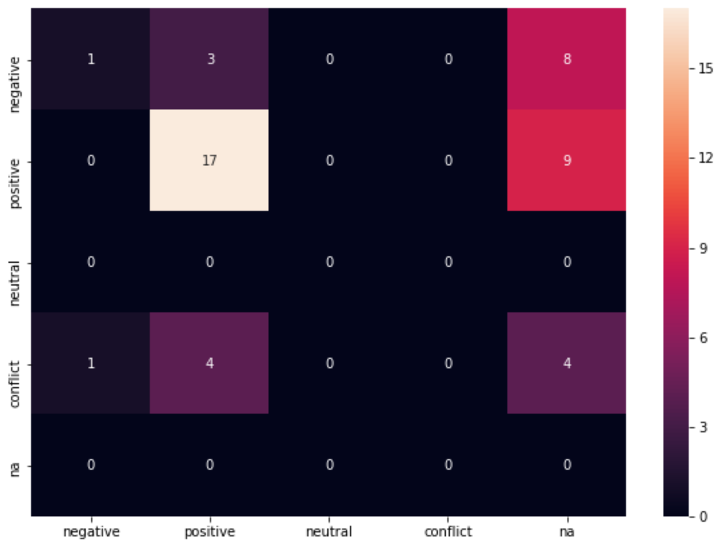


Figure 9: Confusion Matrix: Ambience

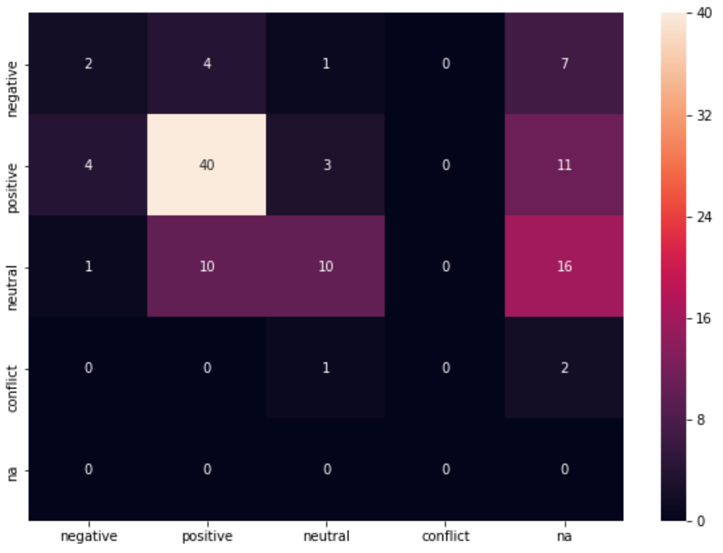


Figure 10: Confusion Matrix: Misc

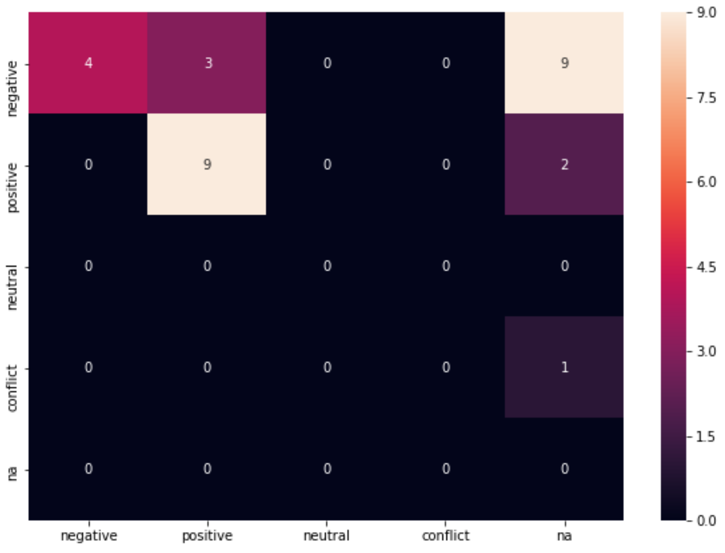


Figure 11: Confusion Matrix: Price

Sentiment Analysis Using PMI Feature

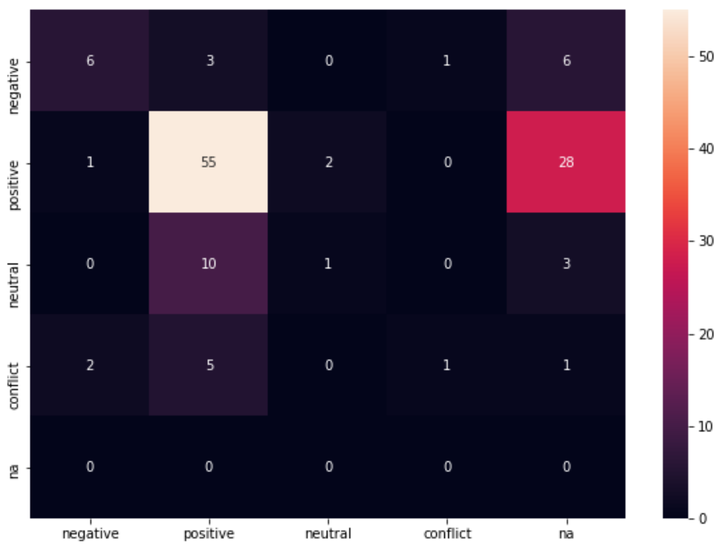


Figure 12: Confusion Matrix: Food

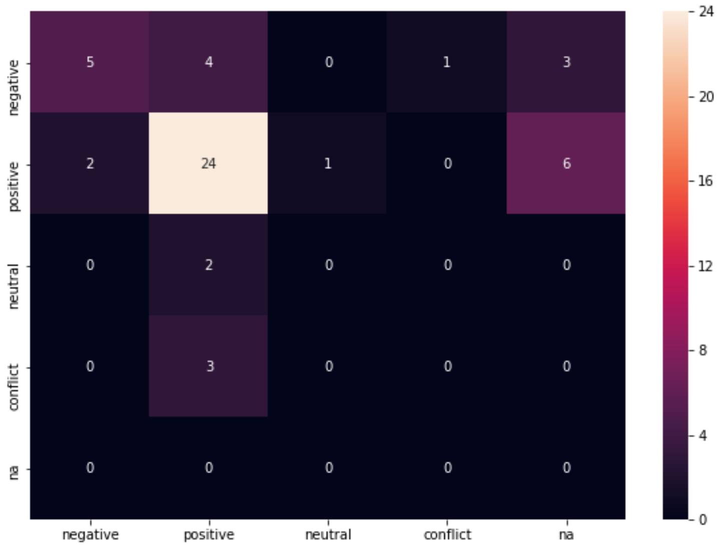


Figure 13: Confusion Matrix: Service

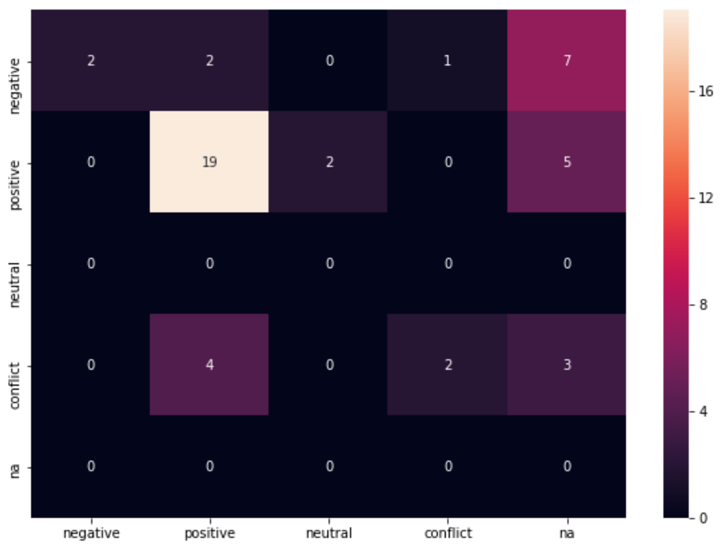


Figure 14: Confusion Matrix: Ambience

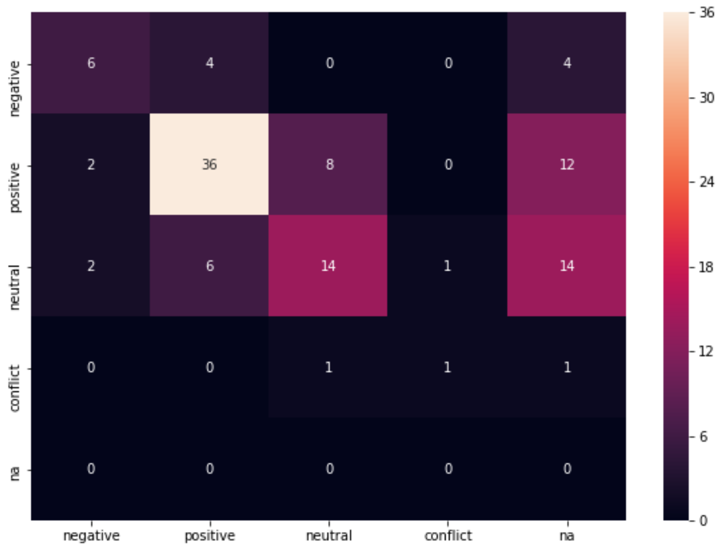


Figure 15: Confusion Matrix: Misc

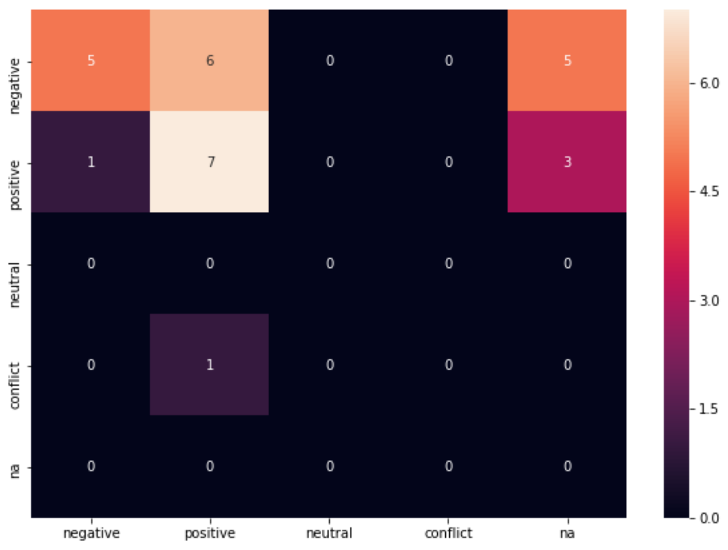


Figure 16: Confusion Matrix: Price