Progress Report I - Neoway Brand Sentiment Recognition Software

Tim Kartawijaya (tak2151), Charlene Luo (cl3788),
Fernando Troeman (ft2515), Nico Winata (nw2408), Jing Yi Zhou (jyz2111)

October 18, 2019

1 Problem Definition and Progress Overview

The Neoway Capstone Project is sponsored by Neoway, a Brazilian data analytics company. Our goal is to create an entity-based sentiment scoring open-source software using cutting edge natural language processing (NLP) packages. One application of our software is to allow companies to gauge consumer sentiments regarding specific brands or products. Companies often gather sentiment data through surveys which provides a limited sample and can be exposed to bias. By using a tool that retrieves sentiment on all entities within a body of text, companies can shape their business and marketing strategy with more data. The example in Figure 1 (provided by Neoway) visualizes how our software would work.

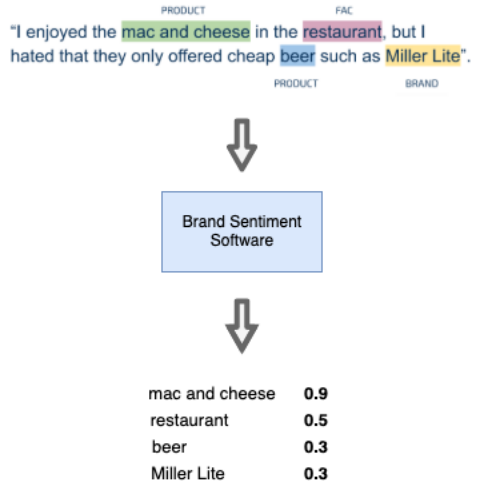

Figure 1: An illustrative example of our software process

Given the sample comment in Figure 1 as an input, we would like to identify that the sentiment towards the product 'mac and cheese' is positive, but the sentiment towards the product 'Miller Lite' beer is negative, and further output sentiment scores on each of these products.

We have made significant progress so far in this project. The data has been cleaned and processed in an ingestible format to allow for convenient analysis regardless of the specific model being used. We've also surveyed the field for the most commonly used NLP packages through academic literature and tested the performance of each package, taking into consideration their speed and effectiveness. Furthermore, we've defined a clear roadmap on our next steps, and prepared a structured repository for our software code (see `https://github.com/timjaya/neoway-brand-sentiment`).

We used Yelp reviews from the Yelp Open Dataset to train and test our model. The dataset is over 10 GB and contains over 15 million rows. As a first step, we will be working with a random sample of 1.5 million rows before training the model on the entire data set.

2 Exploratory Data Analysis

A series of preprocessing steps were conducted on our subset of 1.5 million ratings and reviews, including lemmatization and the removal of stopwords, punctuation and any other unnecessary characters. Complete preprocessing module can be found in the code repository shown above.

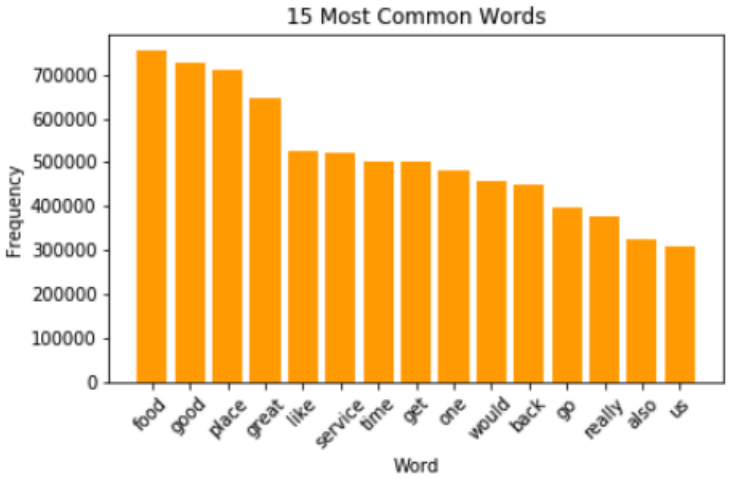

Figure 2: 15 Most Common Words Across All 1.5 Million Reviews

A visualization of the 15 most common words across all text reviews highlighted some interesting insights. 'Food' seems to be, by far, the most-talked about aspect of any particular establishment. 'Service', coming in 6th in the ranking of most common words, seems to be another important determinant of a user's experience.

'Good', 'Great' and 'Like' are some other common terms used across all reviews. This could imply that a majority of our ratings are positive. If so, it would be beneficial to conduct a separate analysis of the most common words in 1-star and 5-star reviews respectively.

There is a good amount of overlap here, with words like 'get', 'food', 'back' appearing in both lists. However, we also observed the presence of certain strictly positive words (e.g. love, amazing, etc.) in 5-star reviews and negative words (e.g. never, etc.) in 1-star reviews.

A bar chart illustrating the distribution of ratings shows us that most reviews are positive (4 stars and above), with almost half of all the reviews in our dataset having a perfect rating of 5 stars. Furthermore, users are rarely neutral about a particular establishment, and any negative sentiment is usually extreme (1 star).

Figures 5 and 6 underline an interesting trend - negative reviews are generally longer in length. Unfavorable reviews are accompanied by lengthy and more detailed descriptions of a user's experience. Great experiences and perfect ratings, on the other hand, are discussed with about 35% fewer words. This particular fact is important to note as it affects our approach on sentiment analysis, particularly regarding the constituency-based parse tree in Section 3.2.3. Since negative reviews are generally longer in length, it would be computationally challenging to analyze these kinds of

Figure 3: Comparison of Most Common Words Between 1-Star and 5-Star Reviews

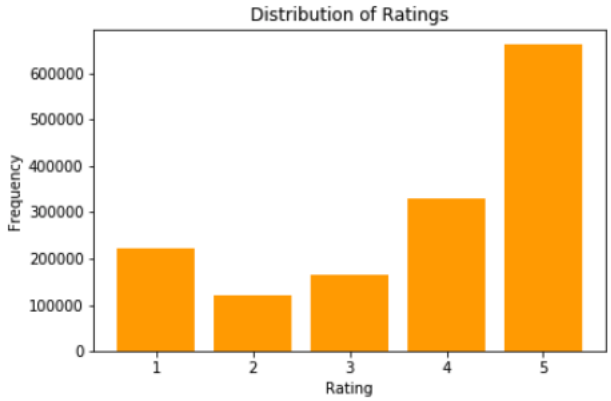

Figure 4: Distribution of Ratings

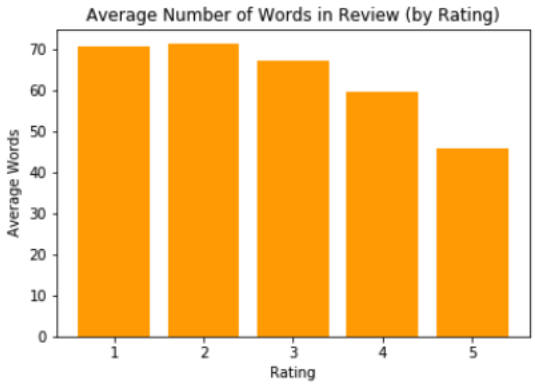

Figure 5: Average Number of Words in Each Review, Grouped by Rating

reviews as our tree would have to grow deeper. A possible solution would be to segment this particular type of review (long in length and overall negative sentiment) then develop an approach that does not have to parse the tree completely. We will further explore this issue as we progress in our projects. See section 3.2.3 for more information on the Parse Tree.

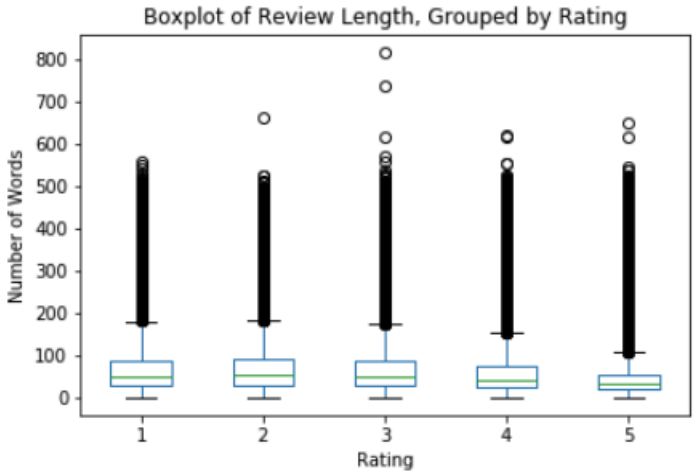

Figure 6: Distribution of Review Length, Grouped by Rating

3 Methodology

3.1 Related Literature

Our goal, as described previously, is to conduct Targeted Sentiment Analysis (TSA): given a review, we want to output all the entities (i.e. products, brands, institutions) mentioned and the sentiment scores towards each of them.

Targeted Sentiment Analysis is a recent problem of interest in data science. Traditionally sentiment analysis and named entity recognition are two distinct areas in NLP, but there is ample recent work on marrying the two. Existing work on these three tasks generally falls into two categories:

1. Using Recurrent Neural Networks: Using a position-aware bidirectional attention network (a type of RNN), it notes both the position of entities in a sentence as well as the relationship between a target entity and its sentence using an attention mechanism. (1).

2. Using Memory Networks: There are methods that can capture sentiments between features that are very far from one another (2). Wang et al. (3) extended the memory mechanisms by adding interaction terms between entities and aspects that might relate to them.

While these are certainly promising methods that can be applied to our problem, their main drawback is they require training our own deep neural network with memory/attention mechanism for TSA. At this stage of the project, we will focus our efforts on leveraging existing pre-trained models to conduct TSA on our dataset. We believe that this is a scalable approach that has shown promise thus far.

3.2 Our Approach

As pretrained models are not readily available for TSA, our solution is to conduct Sentiment Analysis and Named Entity Recognition using separate models, and then join them using a Parse Tree model. Given an input review, we propose the three steps below:

1. **Named Entity Recognition** model will output the list of entities in the review.

2. Given a list of entities, a **parse tree** will determine the **context** surrounding each entity (clauses in the review that is referring to the entity in question)

3. **Sentiment Analysis** model will be used on the **context** of each entity, determining the sentiment score per entity.

The three subsections below will discuss our work on each of these models.

3.2.1 Named Entity Recognition (NER)

From our preliminary comparisons, we identified **SpaCy's NER** as the best model due to its speed and similar performance to more complicated models. In our preliminary tests, all NER models struggle to identify products and brands that are not capitalized. Thus we need to train these NER models further before they can be useful. The comparisons are shown in Table 11. Furthermore, Fig 7 illustrates how NER models without training can only recognize entities based on capitalization. The same results hold for Allen and Stanford NLP even though they are not shown.

```
review = "steak sandwich was delicious, and the caesar salad had an absolutely delicious dressing"
doc = named_entity_recognizer(review)
print([(X.text, X.label_) for X in doc.ents])
```

```
[]
```

```
review = "Steak Sandwich was delicious, and the Caesar Salad had an absolutely delicious dressing"
doc = named_entity_recognizer(review)
print([(X.text, X.label_) for X in doc.ents])
```

```
[('Steak Sandwich', 'PERSON'), ('the Caesar Salad', 'ORG')]
```

Figure 7: SpaCy NER is case-sensitive

Additionally, the current NER models not customized to brands and products, since its main purpose is to detect people, cities, organizations, and so on. Therefore, we need to re-train the model using brand data, which includes developing our own training data. We have made progress in this respect by using WordNet's food dictionary, containing product-words such as bratwurst, beer, etc, which constitutes as our brand list for Yelp Review data (Yelp data rarely contains food and drink brands). We will focus on SpaCy here, as it is very easy to train (providing exact match as well as statistical model options).

3.2.2 Sentiment Analysis

We looked at three different models representing different approaches to sentiment analysis. VADER (5) is a well known industry standard model that uses a rule-based approach. Stanford NLP (4) uses a parse tree along with a deep RNN to better handle complex sentence structures. TextBlob is one of the many naive machine learning approaches that assumes the sentiment of a sentence is the sum of the sentiment of all words. We compared the effectiveness and speed of the three in Table 2.

VADER and Stanford NLP clearly outperforms other models here. Moving forward, we will only consider the two. VADER's advantage lies in its speed, while Stanford NLP is more accurate on complex sentences.

3.2.3 Constituency-based Parse Tree

As discussed, we use a pre-trained parse tree model to determine the context surrounding an entity. The idea of this method comes from studying Stanford NLP's sentiment analysis. Upon obtaining

	SpaCy	Allen NLP	Stanford NLP
Methodology	Feed forward neural net with one hidden layer trained on OntoNote.	Two models provided. The first is a Gated Recurrent Unit (GRU) model. The second is a bi-LSTM-CRF model.	Conditional Random Fields (CRF) model.
Speed (1000 reviews)	20 seconds	6 hours	180 seconds
Pros	(1) Easy to implement and can be trained further. (2) Satisfies most cases tested	(1) Most sophisticated, state-of-the-art of the three compared.	
Cons	(1) Struggles to identify products without further training.	(1) Both models are slow and not as easy to train. (2) Struggles to identify products without further training.	(1) Not as easy to train. (2) Struggles to identify products without further training.

Table 1: Comparison of NER Models

	VADER	Stanford NLP	TextBlob
Methodology	Rule based. Dictionary of words and lexical features with the implied sentiments.	Deep RNN combined with a parse tree.	Naive Bayes model trained on labelled movie reviews.
Speed (1000 reviews)	2 seconds	6 minutes	Less than 1 second
Pros	(1) Fast, light, and easy to implement. (2) Lexical rules perform better than other models that aggregate word sentiments.	(1) Most sophisticated model out of the three. Works well on complex sentence structures (see Fig 8)	(1) Fastest model.
Cons	(1) Still fails to capture complexities that a deep learning model can.	(1) Slow. Requires instantiating local server using CoreNLP API which can complicate implementation.	(1) Aggregates sentiments of words without regard to sentence structure. Cannot detect complex structures.

Table 2: Comparison of Sentiment Analysis Models

a parse tree, we define the **context** of a sentence to be the minimum sentiment-containing sentence (sentences are nodes labelled **S** in Fig 8) that contains the entity.

There are several pretrained models that can be used. We looked at Benepar/Berkeley Neural Parsing (7) and Stanford NLP's models. They have very similar runtimes (1 min 53s [Stanford NLP] and 2 min 33 s [Benepar] on 1000 reviews) and come up with very similar parse trees in the examples we studied.

We show a parse tree obtained using Benepar in Fig 8. The parse tree correctly identifies that

the clause before the first comma is referring to the product Steak sandwich, while everything afterwards refers to Caesar salad (see Figure 8). The sentiment scores of the two contexts are also shown in Figure 9.

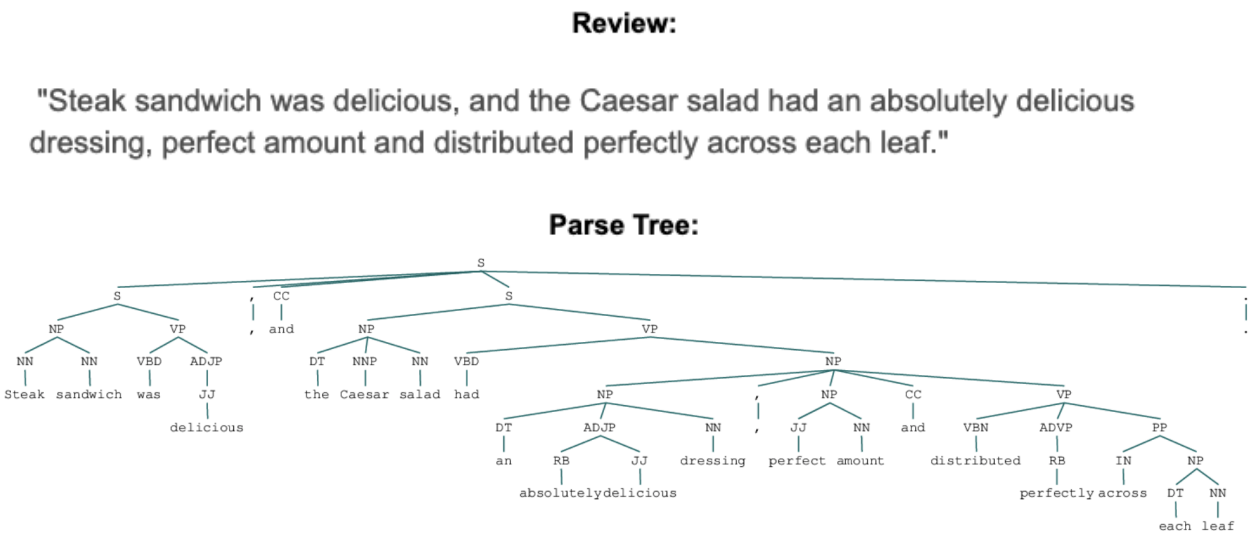

Figure 8: Constituency parsing, example 1. Shown is the text review and the resulting tree structure

In Figure 9, we can see that both VADER and Stanford NLP successfully calculates a higher positive score for Caesar salad than Steak sandwich, which is what a human would do.

However, there are likely to be exceptions to our definition of context. The case in Figure 10 shows an example where we need to traverse further up the tree in order to calculate the correct sentiment score.

There is clearly further work to be done to refine the parsing methodology. For the counterexample in Figure 9, we propose that if the minimum sentence has a neutral sentiment, then we can traverse further until we reach a sentence that has a non-neutral sentiment. We will experiment with different rules and test them (described next section).

3.3 Validation and Testing

As this is an unsupervised learning project, one of our future problems will be how to quantify the performance of one method against the other. Our current plan is to use information from Yelp star ratings to help us validate and compare models.

The idea is as follows. If customers on average love the Caesar salad in a particular restaurant, the reviews mentioning Caesar salad will have a higher average star rating than those that don't. We can use the difference between average star rating of reviews containing an entity and average star rating of the entire population (e.g. average star of a restaurant) to determine the sentiment towards the entity, and scale this metric as appropriate. We can then compare this metric with the sentiment scores generated by various methods to quantify the performance of each method.

4 Goals and Next Steps

We are currently at the 35% mark of our project: we have cleaned and explored the data thoroughly, identified promising methods (Stanford NLP, Constituency Parsing), and determined our approach

Sentiment Scores (VADER):

```
Overall:
Steak sandwich was delicious.the Caesar salad had an absolutely delicious dressing, with a perfect amount of dressing
, and distributed perfectly across each leaf. {'neg': 0.0, 'neu': 0.615, 'pos': 0.385, 'compound': 0.9168}

Sentence1:
Steak sandwich was delicious.---------- {'neg': 0.0, 'neu': 0.448, 'pos': 0.552, 'compound': 0.5719}

Sentence2:
the Caesar salad had an absolutely delicious dressing, with a perfect amount of dressing, and distributed perfectly a
cross each leaf. {'neg': 0.0, 'neu': 0.574, 'pos': 0.426, 'compound': 0.9168}
```

Sentiment Scores (Stanford NLP):

```
Overall:
 Steak sandwich was delicious.the Caesar salad had an absolutely delicious dressing, with a perfect amount of dressin
g, and distributed perfectly across each leaf.

Sentiment: Verypositive
Probability Distribution:
Very Negative,     Negative,         Neutral,          Positive,         Very Positive
[0.0061021498739, 0.00778537690842, 0.10253026211897, 0.42187180321409, 0.46171040788462]

Sentence1:
 Steak sandwich was delicious.

Sentiment: Positive
Probability Distribution:
Very Negative,     Negative,         Neutral,          Positive,         Very Positive
[0.02537623043181, 0.0605944895721, 0.1309570172712, 0.56822740801947, 0.21484485470542]

Sentence2:
 Steak sandwich was delicious.

Sentiment: Positive
Probability Distribution:
Very Negative,     Negative,         Neutral,          Positive,         Very Positive
[0.00093343290854, 0.00080185991799, 0.00841579565213, 0.58314508653382, 0.40670382498752]
```

Figure 9: Sentiment analysis score results using Stanford NLP and VADER from example 1

Review:

"Ordered strawberry açaí and it was a little too sweet."

Parse Tree:

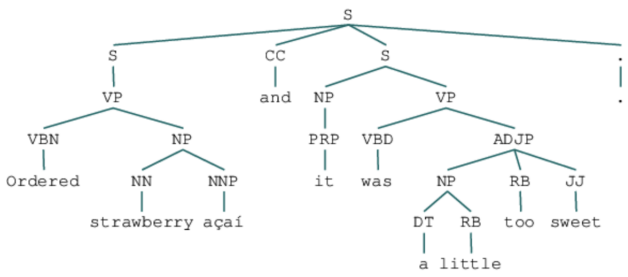

Figure 10: Constituency parsing, example 2. Shown is the text review and the resulting tree structure

in integrating these models together. Our roadmap will be based around the structure of our software process, as shown in Figure 12.

Sentiment (VADER):

```
Overall:
Ordered strawberry açaí and it was a little too sweet. {'neg': 0.0, 'neu': 0.746, 'pos': 0.254, 'compound': 0.4062}

Sentence1:
Ordered strawberry açaí----------------- {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

Sentence2:
it was a little too sweet.-------------- {'neg': 0.0, 'neu': 0.595, 'pos': 0.405, 'compound': 0.4062}
```

Sentiment (Stanford NLP):

```
Overall:
 Ordered strawberry açaí and it was a little too sweet.

Sentiment: Negative
Probability Distribution:
Very Negative,     Negative,         Neutral,          Positive,         Very Positive
[0.1785988276438, 0.61755240542823, 0.1510418169514, 0.03179869659471, 0.02100825338187]

Sentence1:
 Ordered strawberry açaí

Sentiment: Neutral
Probability Distribution:
Very Negative,     Negative,         Neutral,          Positive,         Very Positive
[0.04102559690917, 0.2465292928676, 0.47896107970388, 0.19671791256638, 0.03676611795297]

Sentence2:
 it was a little too sweet.

Sentiment: Negative
Probability Distribution:
Very Negative,     Negative,         Neutral,          Positive,         Very Positive
[0.10553174755899, 0.55505525670377, 0.24676427271945, 0.07297091804494, 0.01967780497286]
```

Figure 11: Sentiment analysis score results using Stanford NLP and VADER from example 2. Note that VADER fails to calculate the correct sentiment score here

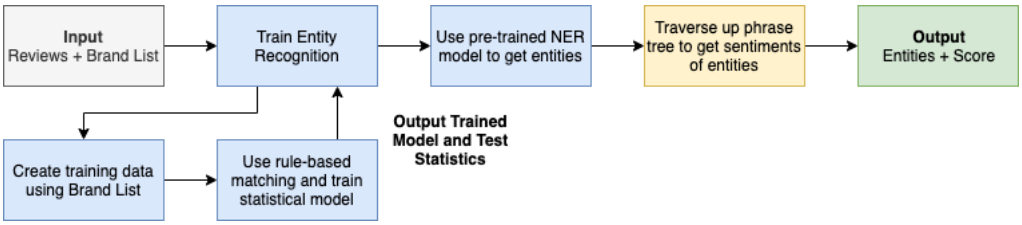

Figure 12: Initial design on software process.

The structure above is derived from our approach in Section 3.2. From this structure, our goals and next steps is summarized below:

1. **Train Brand-Sensitive Entity Recognition:** As mentioned, we need to train SpaCy before it can be usable on the Yelp dataset. We will aim to make the training process automated and generalizable, so that Neoway can use our software easily, only needing to input their own brand list and reviews.

2. **Improve Parsing and Sentiment Analysis Model:** The current parsing methodology and sentiment analysis has proven to work, but is not perfect. The next step is to compare different parsing rules using the validation rules discussed to choose the best rule and model.

3. **Continue Literature Research:** There are still models and articles that may be relevant to our project and we will continue to review and integrate ideas that improves our approach.

4. **Optimize Prediction Speed:** Our best performing sentiment analysis model, Stanford NLP, is quite slow in practice, as shown in Table 2. We are seeking to find ways to optimize the speed of this model through parallelization, which is possible for Stanford NLP.

5. **Productionization and Documentation:** After reaching acceptable performance of the models above, we then will integrate all the components together into a cohesive open-source package. We have set a few weeks to solely focus on software development, making sure our process is robust to various data inputs and has reproducible results through unit testing and utilizing Docker/Travis CI.

5 Contribution

- **Charlene Luo:** Main contributor on research and literature review of entity recognition models; tested and compared said models.
- **Nico Winata:** Main contributor on research and literature review on sentiment analysis and parse tree models; tested and compared said models.
- **Jing Yi Zhou:** Cleaned and preprocessed data. Main contributor in installation and implementation of each model, comparing efficiency/speed, exploring parallelization methods.
- **Fernando Troeman:** Cleaned and preprocessed data. Main contributor on exploratory data analysis on cleaned data and contributed to software structure.
- **Tim Kartawijaya:** Team Captain: set up meetings, milestones, and manage progress. Contributed to data cleaning and main contributor in designing and preparing software structure.

References

[1] Shuqin Gu, Lipeng Zhang, Yuexian Hou, and Yin Song. 2018. A Position-aware Bidirectional Attention Network for Aspect-level Sentiment Analysis. *In COLING. 774–784.*

[2] Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective LSTMs for Target-Dependent Sentiment Classification. In *COLING. 3298–3307.* Available at `https://github.com/ganeshjawahar/mem_absa`

[3] Shuai Wang, Sahisnu Mazumder, Bing Liu, Mianwei Zhou, and Yi Chang. 2018. Target-Sensitive Memory Networks for Aspect Sentiment Classification. In *ACL.*

[4] Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013b. Recursive deep models for semantic com- positionality over a sentiment treebank. *Conference on Empirical Methods in Natural Language Processing.*

[5] C. J. Hutto, Eric Gilbert. 2014. VADER. A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. *International AAAI Conference on Web and Social Media, North America, may. 2014.*

[6] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li, Incorporating Copying Mechanism in Sequence- to-Sequence Learning. *CoRR, abs/1603.06393, 2016.*

[7] David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? An analysis. In *Association for Computational Linguistics.*