Final Report - Neoway Brand Sentiment Recognition Software

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

December 21, 2019

Abstract

This served as background

We have developed a sentiment analysis software for Neoway that is able to identify and generate sentiment scores for individual entities in any given review. Often, sentiment analysis is generated on entire blocks of text - an overall sentiment is attached to each review. We take it one step further by isolating the sentence context and sentiment of the entities. The resulting software employs entity recognition (e.g. SpaCy), constituency parsing and sentiment analysis models (e.g. VADER and Stanford NLP) to perform the stated task. Results were validated on an original order-ranked validation process using a subset of 15,000 reviews and indicate satisfactory performance.

1 Introduction

Overview of approach

This Capstone Project is sponsored by Neoway, a Brazilian data analytics company. Our goal is to create a **Targeted Sentiment Analysis (TSA)** software. The software will parse a user's review of a commercial establishment, identify relevant entities, and create a sentiment score associated with that entity. The business application of such software is to allow companies to gauge consumer sentiments regarding specific brands or products. Here, our specific reviews are Yelp restaurant reviews from the Yelp Open Dataset. Figure 6 (provided by Neoway) demonstrates the software process.

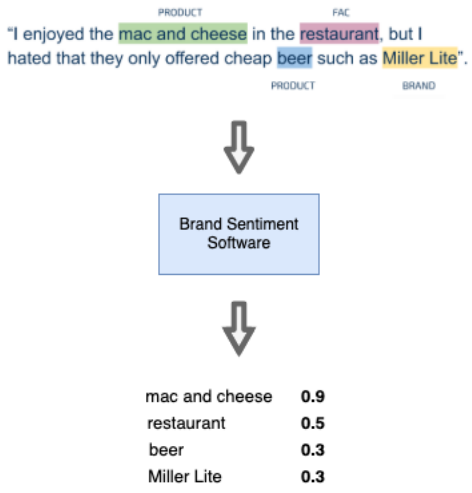

Figure 1: An illustrative example of our software process

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.

2 Dataset and Exploratory Data Analysis

The Yelp dataset is over 10 GB and contains over 15 million rows, and we extracted a subset of 1.9 million ratings, containing only reviews of restaurants. A series of preprocessing steps were conducted on our subset, including lemmatization and the removal of stopwords, punctuation and any other unnecessary characters. The complete preprocessing module can be found in our project's Github repository (1).

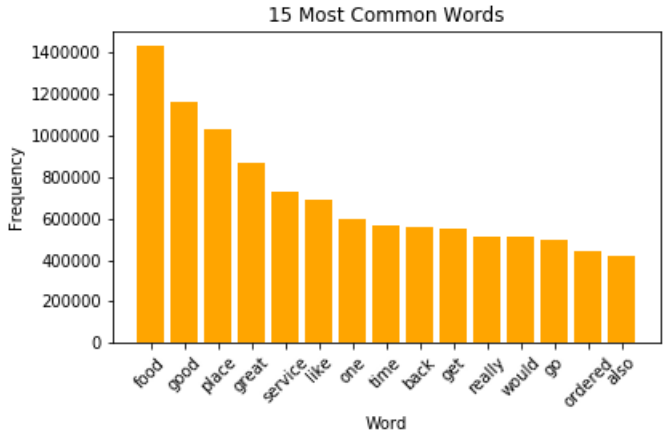

Figure 2: 15 Most Common Words Across All 1.9 Million Reviews

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 5th 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.

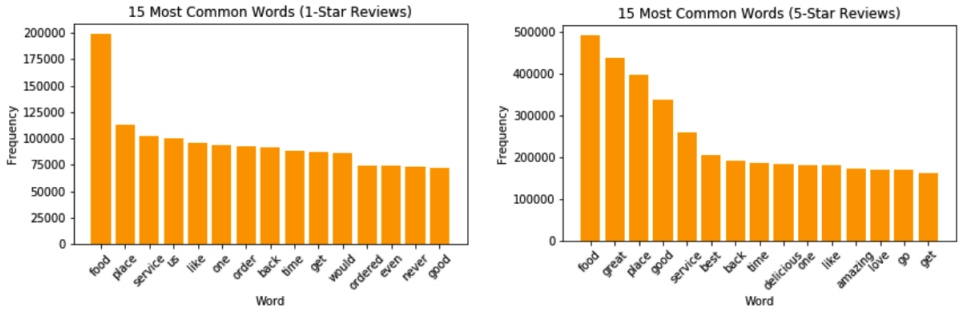

Figure 3: Comparison of Most Common Words Between 1-Star and 5-Star Reviews

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.

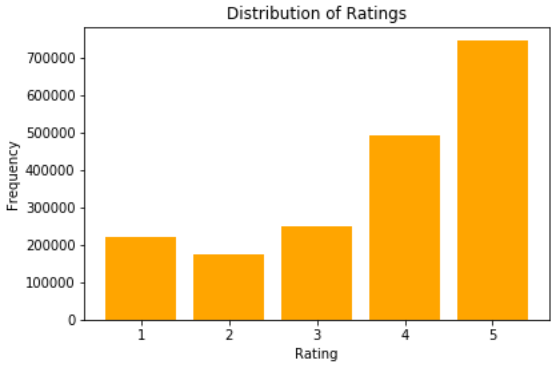

Figure 4: Distribution of Ratings

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).

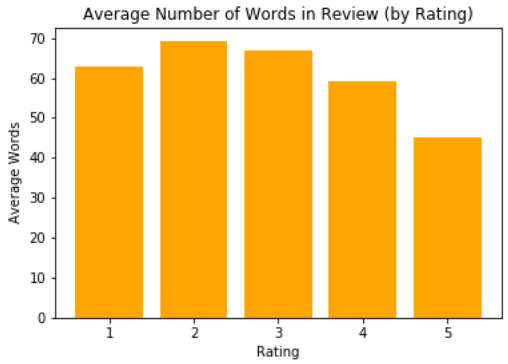

Figure 5: Average Number of Words in Each Review, Grouped by Rating

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 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) and then develop an approach that does not have to parse the tree completely.

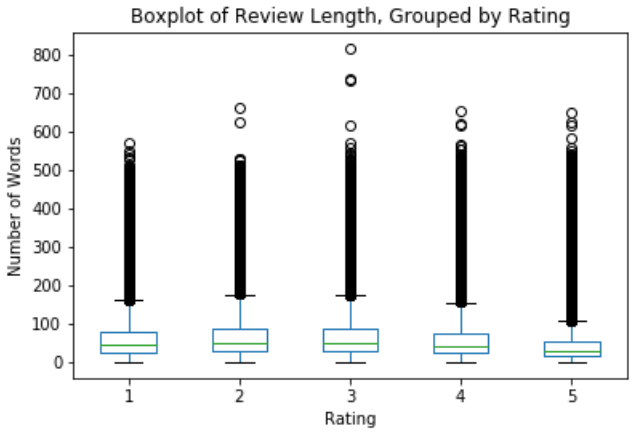

Figure 6: Distribution of Review Length, Grouped by Rating

3 Related Work

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 generally falls into two categories:

1. Using Recurrent Neural Networks and attention mechanisms: A recurrent neural net and its variants are able to learn the interactions between entities and words describing them, for example by using a latent attention vector. For example, Gu et al. used a position-aware bidirectional attention network to note both the position of entities in a sentence as well as the relationship between a target entity and its sentence using attention mechanism. (5).

2. Using Memory Networks: These are methods that can capture sentiments between features that are very far from one another, examples include (6) and (7) which improves sentiment analysis performance from traditional models. Wang et al. (8) extended the memory mechanisms by adding interaction terms between entities and aspects that might relate to them, hence applying Memory Networks to the task of TSA.

Our approach differs from the papers discussed above. Instead of constructing a complex model that solves our task, we show that we can use a combination of several existing pre-trained models in a particular way to solve TSA. We believe that this is a scalable approach that has shown promise in our test cases. It is highly applicable to business settings and its generality means it does not require significant retraining when applied to a different dataset. An added benefit (as we will see in the next section) is that our approach allows us to output the parts of a review that discusses each entity we are interested in. This is potentially very useful for a business setting.

4 Modeling

4.1 Overview

Our proposed solution comes in three steps:

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

2. Given a list of entities, a **parser** 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.

4.2 Entity Recognition

From our preliminary comparisons, we identified **SpaCy's NER** as the best model due to its speed and similar performance to more complicated models. The comparisons between different NER models are shown in Table 1. A SpaCy model (3) is not simply trained to 'memorize' hard-coded entities, but rather made to understand what constitutes an entity in any given context. In other words, it is a model that can be generalized across other examples and detect previously unseen entities.

In our preliminary tests, SpaCy's default NER models struggle to identify products and brands that are not capitalized. 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 are not customized to brands and products, since its main purpose is to detect people, cities, organizations, and so on. Therefore, to solve the two issues above, we need to re-train the model using brand data, which includes developing our own training data. The focus will be upon SpaCy , as it is efficient to train and provides multiple model options: exact match and statistical model.

4.2.1 Training the Spacy Model

Training a SpaCy model requires training data that conveys our definition of entities. As our goal is to track consumer sentiment towards separate entities within restaurant reviews on Yelp, the entities we are looking for are food and beverage products (e.g. french fries, orange juice, etc.). Depending on the use case, we will also be able to generalize this model to include specific brand names for sentiment analysis on brands and companies (e.g. Pepsi, Budweiser, etc.).

To train a Spacy model, the target label of the training data must be in a specified format. An example of training data appropriate for Spacy is shown in Figure 8. For each review, we will note the exact position (starting and ending indices) of relevant entities, as well as the type of entity identified i.e. "PRODUCT" in our case. Given this manually annotated training set, the Spacy model attempts to learn to identify entities within the context of each review.

Implementing the training comes in two steps.

```
train_data = [
    ("The chicken was overcooked but the pork was delicious", [(4, 10, 'PRODUCT'), (35, 38, 'PRODUCT')]),
    ("We would definitely recommend this cafe!", [(35, 38, 'ESTABLISHMENT')])
]
```

Figure 8: Example of Generated Spacy Training Data

Firstly, we collect a list of relevant food and beverage products that we would like to define as entities. The product list used for this project is largely made up of food and beverage entities derived from WordNet - a large lexical database of English. It consists of food products, ranging from generic ingredients such as 'fish' and 'chicken' to more specific dishes like 'Manhattan Clam Chowder'. We also manually added several words to the product list - including terms used to describe a particular establishment (e.g. restaurant, diner, cafe, etc.) - this will prove useful in the next steps, allowing us attach sentiment to the establishments themselves.

Secondly, using this product list, we write code that labels the positions and type of matched entities in the reviews found in our Yelp data set. (see Figure 8) Our algorithm runs through every review in the Yelp data set, identifying matching entities and labelling their exact positions within the text. The resulting training data, specifically optimized using Yelp reviews, is then used to train our Spacy model in recognizing entities in other, unseen Yelp reviews.

	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

4.2.2 Entity Recognition Results

The reviews were split into a train and test set with 80% / 20% ratio. In addition to creating our own entities list with the "Product" label (as described in Section 4.2.1), we also add on Spacy's default entities that have labels such as Cardinal, Money, and Date. The reason we perform this extra step is due to Spacy's tendency 'forget' the previously learned entities when learning a new entity after a few iterations. More on this subject can be found in (4). Appending the default Spacy entities onto our training set guarantees that our model 'remembers' the entities that it already

trained on.

The predictions of our ER model achieved a test recall of **91.4%** and test precision of **90.7%**, meaning our model was able to detect 91.4% of the true entities (both Spacy's default and our custom entities) in the test set, and 90.7% of the predictions are true entities.

In addition to the metric above, we also analyzed how well the model detects entities it has never seen before i.e. its novelty metric. We first extracted a list of product entities that existed both in the list of entities in the training set as well as the list of entities in the validation set. Half of these entities are then removed, i.e. masked, from the train and validation data set. Then, the model was trained in two separate experiments: once with the original unmasked training set, and once with masked. We then compared the ratio of the number of appearances of an entity in predictions divided by the number of appearances in true labels for that review, averaged over reviews. 75% of masked entities had the same number of appearances in the prediction regardless of masking; on average, the unmasked novelty ratio was 2.83% higher than that of the masked experiment. By comparing these ratios, we were able to examine the the performance of the model in predicting labels that are not in its training set. From this experiment, the model displayed its ability to label entities that are not present in its training set since removing half of the labels from the training set had a very small effect on the predictions from the model.

4.3 Parsing and Sentiment Analysis

After obtaining a list of entities per review, the next step is to determine the context. We focused on using constituency parsing, an approach that is inspired by (9). There are other methods of context determination as well, for example using dependency parsing in (14) and word embeddings in (13). We also applied dependency parsing, but with a smaller degree of success.

The primary benefit of our approach is that using full sentences as contexts (as opposed to individual adjectives, additive word embeddings) allows for a more complex sentiment analysis. The example we gave in the previous report is the review *"Ordered strawberry açai. It was a little too sweet"* where the sentence contains no instance of negative words but expresses a negative sentiment. A word-based sentiment analysis would find this sentence challenging, but complex sentence-based sentiment analysis can resolve this issue (VADER, Stanford NLP).

4.3.1 Constituency Parsing Rules

Upon processing a review with a constituency parsing algorithm (the benepar package (12)) , we obtain a parse tree describing the syntax of the review. Next, we determine the context surrounding a tree using a **parsing rule**. We have explored several rules to determine the context surrounding an entity, described in Table 2.

The rationale behind each rule is as follows.

- Rule 1 is the most simple way to use the parse tree, namely relying on the parse tree to identify full sentences surrounding each entity. A full sentence here is a clause that can stand as its own independent sentence, for instance *"The fried chicken was delicious but the service was bad"* would be split into two sentences *"The fried chicken was delicious"* and *"but the service was bad"*. The labelling here is conducted by an off-the-shelf constituency parsing model called Benepar (12).

- Rule 2 is motivated by observations that minimum sentences are often inadequate. An example of this is *"Ordered strawberry açai. It was a little too sweet."*. In this case some parsers

Rule	Description
1	Context is defined as the minimum sentence (S-labelled node) containing the entity.
2	Assign contexts using rule 1. While context's sentiment is *neutral*, replace with the next minimum sentence.
3	Take the average sentiment of all sentences (S-labelled nodes) containing the entity.
4	Context is initially defined as the minimum sentence containing the entity. If there is a portion of review not assigned to an entity, assign to the nearest previous entity.
5	Assign contexts using rule 1. If the sentiment is *neutral*, use rule 4
6	(Without parse tree) Split the review by punctuation marks (.?!), assigning each chunk to the last entity mentioned in the review.

Table 2: Descriptions of the different constituency parsing rules for sentiment analysis.

would output *"Ordered strawberry açai."* under Rule 1 - its selection would be too short. The neutrality is often an indication of an inadequately long sentence, which is why we use sentiment to determine when to stop.

- Rule 3 is self-explanatory.
- Rules 4 & 5 has the same motivation as Rule 2, however without relying on S-node labelling by the parsing algorithm. Instead, it uses the observation that sentences in the review that doesn't contain an entity often talks about the previous mentioned entity, e.g. *"It was a little too sweet."*
- Rule 6 is a completely parse-tree independent rule, using the same technique as in Rule 4. This rule acts as a baseline for us in comparing the other parsing algorithms.

4.3.2 Dependency Parsing Rules

The way we can use a dependency parser to determine contexts is as a sentence splitter - a dependency parsing algorithm (SpaCy package (3)) splits an input text into separate chunks each containing a root node and other nodes that recursively point towards the root. We can consider each connected chunk as a sentence. The rules we explored are below. Their motivations are like rules 1 and 4 in the previous section respectively.

Rule	Description
7	Context is the minimum sentence (dependency parser chunk) containing the entity.
8	Apply rule 4 on dependency parser sentences.

Table 3: Descriptions of the different dependency parsing rules for sentiment analysis.

4.3.3 Sentiment Analysis

From exploring different sentiment analysis algorithms, we found two most promising pretrained models: VADER and Stanford NLP. The comparison can be found in table 4. The examples in this section are done using VADER and further sections will show VADER having better performance for the task at hand (Section 5).

	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 4: Comparison of NER Models

Qualitative results are important for readers to get a sense of how the method works and to show strengths and weaknesses that may not be captured by metric

4.3.4 Qualitative Results

A qualitative comparison of the different rules is provided below.

REVIEW: "We stopped here for a quick bite before heading to another spot and we stumbled upon their $2 Taco Tuesday. Although you cannot select any taco from the menu the choices provided are quite good. I had 3 tacos the Standard which was my favorite, the pork was okay and the carne was typical, no surprises. I have to say that the Prickly Pear margarita was the absolute best!!!!!"
ENTITIES: ['pork', 'margarita', 'taco', 'menu', 'bite', 'pear']

Detailed results and the calculated sentiment scores (VADER) is provided in Table 2. We selected the three entities to illustrate where each rule excels.

In general, rules 1 and 2 have decent to good performance (also shown in the next section), generally selecting appropriate contexts without including redundant sentences. Its weakness (see Table 2 and Figure 2) is in its reliance on the parsing algorithm - sentences can often be labelled poorly by the parsing algorithm especially if it is not well punctuated, e.g. "3 tacos the Standard which was my favorite" is labelled as a sentence. Another notable problem is that the parsing algorithm sometimes does not append punctuation marks as part of the original sentence.

The performance of Rule 3 is best quantified in the next section, but here we see an overly positive score for entity 'pork'. Rules 4 and 5 often lead to better formed sentences than rules 1 and 2. However, they attach additional portions of the review that are not captured and these are often redundant, as seen in the 'Pork' entity in Figure 2.

Rule 6 is a method that does not use parse trees. It is significantly faster than the other five rules, where parsing takes up most of the computation time. The selected context is often longer

than the other rules, and can contain redundancies. However, Figure 2 shows that the selections are more often than not reasonable. If computation time is a concern, rule 6 is an excellent alternative to using parsing. The sentiment score for Margarita is due to an bug within the VADER software, where more than three consecutive exclamation marks is processed as negative sentiment.

Rule 7 and Rule 8 both use dependency parsing. Qualitatively, we see more errors - the context for pork is too long, for instance. We argue that this is a benefit of constituency parsing over dependency parsing: a constituency parsing sentence is a minimum clause, while a dependency parsing sentence is a string of connected words which can get quite long as Table 6 suggests.

Entity	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
Pork	the pork was okay	the pork was okay		the pork was okay and the carne was typical , no surprises . I have to say that	the pork was okay and the carne was typical , no surprises . I have to say that	I had 3 tacos the Standard which was my favorite, the pork was okay and the carne was typical, no surprises.
	0.226	0.226	0.632	0.153	0.153	0.557
Margarita	the Prickly Pear margarita was the absolute best	the Prickly Pear margarita was the absolute best		the Prickly Pear margarita was the absolute best ! ! ! ! !	the Prickly Pear margarita was the absolute best ! ! ! ! !	I have to say that the Prickly Pear margarita was the absolute best!!!!!
	0.511	0.511	0.673	0.667	0.667	-0.471
Taco	3 tacos the Standard which was my favorite	3 tacos the Standard which was my favorite		3 tacos the Standard which was my favorite ,	3 tacos the Standard which was my favorite ,	Although you cannot select any taco from the menu the choices provided are quite good.
	0.459	0.459	0.513	0.459	0.459	0.493

Table 5: Comparison of different rules and results (Constituency Parsing). Green cells highlight the best rule for each entity, while red cells highlight the notable errors.

Entity	Rule 7	Rule 8
Pork	I had 3 tacos the Standard which was my favorite, the pork was okay and the carne was typical, no surprises.	I had 3 tacos the Standard which was my favorite, the pork was okay and the carne was typical, no surprises.
	0.557	0.557
Margarita	I have to say that the Prickly Pear margarita was the absolute best!!!!!	I have to say that the Prickly Pear margarita was the absolute best!!!!!
	-0.471	-0.471
Taco	I had 3 tacos the Standard which was my favorite, the pork was okay and the carne was typical, no surprises.	I had 3 tacos the Standard which was my favorite, the pork was okay and the carne was typical, no surprises.
	0.557	0.557

Table 6: Comparison of different rules and results (Dependency Parsing). Red cells highlight the notable errors.

In conclusion, we can see that rules 1, 2, and 6 often give out reasonable contexts, with rule 2 taking up the most computational power and rule 6 taking up the least. Rules 4 and 5, 7 and 8 seem initially promising but in practice adds redundancies to the selected contexts. This finding will be further corroborated in the next section.

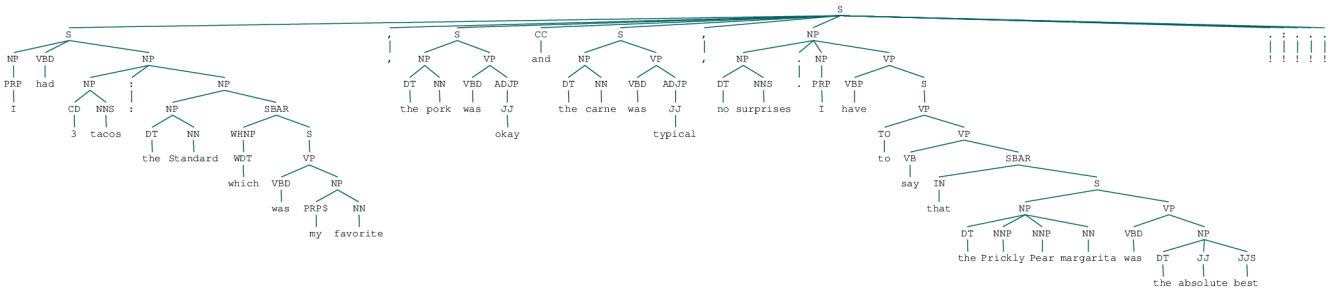

Figure 9: A showcase of parsing (Benepar package) on the sentence "I had 3 tacos the Standard which was my favorite, the pork was okay and the carne was typical, no surprises. I have to say that the Prickly Pear margarita was the absolute best!!!!!". Note that there are mislabelled sentences, and punctuations are often not part of S-labelled nodes.

5 Validation

In the previous section, we have presented different possible methods to conduct parsing and sentiment analysis. The final step is to run these models end-to-end, and then comparing their results. This allows us to choose between parsing rules and sentiment analysis models.

A challenge in doing such validation is the lack of target labels in the dataset. Each review do not have a "true" set of products with associated sentiment scores, thus we cannot quantitatively validate the models without developing labels manually.

To circumvent said obstacle, we developed a ranking-based validation method that uses Yelp stars, a feature we have associated with each review, as a proxy to determine the population's sentiment toward an entity. The process is illustrated in Figure 10 and is as follows:

1. Subset reviews to only originate from restaurants with similar overall ratings (i.e. only use reviews from 3-4 star restaurants) and enough reviews ($\geq$ 1000 reviews). Filtering ratings is a way to remove the effect of the overall restaurant reputation.

2. For each restaurant calculate its ranking score:

   (a) Determine the entities/products existing in the restaurant using the entity recognition model.

   (b) Only consider entities that have enough reviews in that restaurant. For our case, each entity is filtered have 30 reviews. Threshold is set at 30 to ensure a balance between each restaurant's entity list and each entity's number of reviews.

   (c) For each of those entities, calculate two metrics:

      i. The "true" sentiment score, which is the average Yelp stars/rating of reviews that mention the specific entity (only in that particular restaurant).

      ii. The "predicted" sentiment score, which is calculated by using our NLP tool to calculate the score for each review's entities.

   (d) Rank the entities, first by true scores and then predicted scores to two lists.

   (e) Calculate the Spearman's rank coefficient/score of those two lists.

3. Average the rank correlation score over all restaurants.

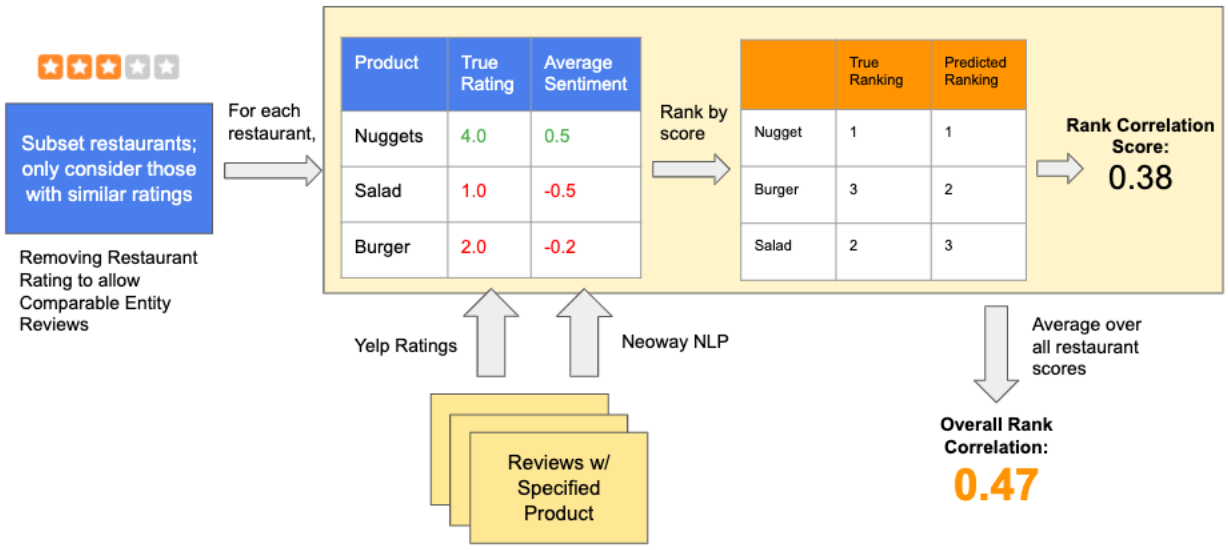

Figure 10: Illustration of end-to-end validation process.

The final rank correlation score will then determine how well our overall process agrees with the given Yelp stars, which is an approximation of how the population feels regarding each entity. Note that we use an ordinal methodology since the predicted and true scores are not directly comparable.

It is possible that for a single review, its Yelp stars are an inaccurate reflection of a product's sentiment. For example, a comment containing 'taco' is 1 star, but actually only negatively talking about the 'service'. Our assumption is that if there are enough reviews for an entity (i.e. ≥ 30 reviews for tacos), then the overall sentiment from the Yelp stars should be reasonably accurate. Therefore, a good result do not have to be a perfect correlation score of 1, but a score that show that both rankings agree moderately. We use this number to complement our qualitative analysis shown in previous section.

5.0.1 Comparing Sentiment Analysis Models

First, we use the method above to compare between Stanford NLP and VADER as our sentiment models, using Rule 6 of our parse (our baseline parsing rule). Stanford NLP's overall score was **-0.192** while Vader's was **0.2535**. We previously thought that Stanford NLP was able to handle more complex sentences, however this result show that it performs *worse* on average. We show on table 3 examples of VADER outperforming Stanford NLP - highlighting the latter's *overconfidence* and tendency to predict negative on a neutral comment. Conversely, VADER's tendency to predict 0 for sentences not clearly positive/negative is desirable here and makes it interact particularly well with Rule 2, as seen in the next section. Another possible reason is that VADER's rules are informed by social media use (e.g. tweets).

5.1 Comparing Parsing Rules

Using VADER, we then compared the six different parsing rules by performing the above process on 10 different restaurants, containing about 15,000 reviews. We show two scores for each rule: a score over all restaurants and a score over restaurants with a high variety in its entities' true ratings (8 restaurants). Due to the ordinal nature of our method, if a restaurant has approximately the same

Review	Stanford		VADER
I got the chicken burrito which looked unassuming but was full of flavor and huge. My wife got the kale salad and it was awesome. We will definitely be back !	Very Negative	0.13	0.93
	Negative	0.59	
	Neutral	0.20	
	Positive	0.06	
	Very Positive	0.02	
My husband and I had some errands to run in Gilbert and We drove by this place so We thought we would give it a try.	Very Negative	0.32	0.00
	Negative	0.60	
	Neutral	0.06	
	Positive	0.01	
	Very Positive	0.01	

Table 7: Comparing VADER and Stanford NLP.

ratings on all its entities, the correlation score can be highly volatile and affect our interpretation negatively. Therefore, the second score will give a cleaner picture on the parsing rule performance. Results can be seen on Table 8.

Rule 6 acts as a baseline here, being completely punctuation based. Constituency parser rules (rules 1 to 5), improve the context detection, with Rule 2 having the best correlation score. In general shows the strength of the constituency parser in selecting relevant context as compared to other approaches which often include redundancies. Despite its downside in terms of processing speed, due to its strengths we will use Rule 2 for our production code. It is interesting to see that a punctuation based approach (rule 6) outperforms dependency parsing (rules 7 and 8). This corroborates our observation in the previous section that a dependency parser's connected components contain too long a sentence with potentially sentiment indicators for other entities.

Rules	All Restaurants Score	High Variety Restaurants Score	Total Processing Time
Rule 1	0.23	0.25	1h 57m
Rule 2	**0.44**	**0.58**	2h 12m
Rule 3	0.41	0.51	2h 2m
Rule 4	0.32	0.40	1h 20m
Rule 5	0.20	0.23	1h 56m
Rule 6	0.19	0.24	**12m**
Rule 7	0.00	0.07	14m
Rule 8	0.10	0.14	33m

Table 8: Spearman rank correlation scores for each parsing rule and process time for each rule (15000 reviews, 10 restaurants). Bold indicates best in category.

6 Software

The methodology above is then implemented as a Python package in an open-source repository. Our code structure is based on Neoway's Data Science code template (15), which provides built-in documentation and folder structure. Our code design provides modular code to preprocess data, train, predict, and output metadata i.e. training validation metrics. Included in the repository are Jupyter notebooks to run the end-to-end validation process above. Code and documentation on how to use said packages can be found in the provided Github link below (1).

7 Conclusion

In conclusion, we developed an open source Python software to run targeted sentiment analysis on the Yelp Open Dataset. To achieve this, we developed a method of combining pretrained NLP packages (ER, parsing, and sentiment analysis) to carry out our task.

Novel methods to find the correct context of an entity are developed and validated using our own ordinal-based validation metric, which takes advantage of the fact that we have star ratings as a quasi-label. This has helped us decide and improve on models in a systematic way. Our qualitative and quantitative results were satisfactory in each modeling step (NER and Sentiment Analysis) and also as a cohesive, end-to-end process.

8 Future Work

Our validation metric can be extended to compare our performance against other state of the art models that we've previous mentioned such as attention networks (5) or memory networks (8). In this way, we can benchmark our application-based approach against state of the art approaches in academic literature.

There are also aspects of our current methodologies that can be further honed. An example of this is how we are only using connected components of a dependency parse graph. We can build a more sophisticated parsing rule by taking advantage of the labelled edges in a tree.

In regards to software, there is more room for improvement in regards to efficiency. The current bottleneck of our software is the implementation of the rules, specifically in the constituency parsing algorithm itself (Benepar package). The algorithm does not scale well to very long reviews, as larger trees would need to be created. This is particularly challenging to resolve as we would need to find parsing software alternatives or determine a way to parallelize Benepar.

References

[1] Neoway Columbia Data Science Team. "Neoway Sentiment Analysis Software." github.com, https://github.com/timjaya/neoway-brand-sentiment. Accessed 20 December 2019.

[2] Neoway Labs Team. "Data Science Template." github.com, https://https://github.com/NeowayLabs/data-science-template. Accessed 25 November 2019.

[3] Matthew Honnibal and Ines Montani. 2018. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

[4] Honnibal, Matthew. "Pseudo-rehearsal: A simple solution to catastrophic forgetting for NLP." *explosion.ai, https://explosion.ai/blog/pseudo-rehearsal-catastrophic-forgetting.* Accessed 25 November 2019.

[5] 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.*

[6] 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`

[7] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pgs 2440–2448.

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

[9] 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.*

[10] 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.*

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

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

[13] Youngmin Lee, Seula Park, Kiyun Yu, and Jiyoung Kim. 2018. Building Place-Specific Sentiment Lexicon. In Proceedings of the 2nd International Conference on Digital Signal Processing (ICDSP 2018). ACM, New York, NY, USA, 147-150. DOI: https://doi.org/10.1145/3193025.3193050

[14] Yasavur, U.; Travieso, J.; Lisetti, C.; Rishe, N.. Sentiment Analysis Using Dependency Trees and Named-Entities. Florida Artificial Intelligence Research Society Conference, North America, may. 2014. Available at: ¡https://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS14/paper/view/7869¿.

[15] Neoway Labs Team. "Data Science Template." github.com, https://https://github.com/NeowayLabs/data-science-template. Accessed 25 November 2019.