Individual Report on Project (10): Movie Review Analysis

Name: Vitid Nakareseisoon

Email address: nakarese@usc.edu

Rest of my group:

- Minu George
- Soumya Ravi
- Tanshi Sharma

URL of GitHub repository:

As the consolidated repository is preferred, I uploaded my works to https://github.com/TanshiSharma/NLP .

The original repositories (which contain a proper series of commits and descriptions) can be found here:

- Aspect-Sentiment Mapping: https://github.com/vitid/SentimentExtractor
- ScrapyIMDB (Review Crawler): https://github.com/vitid/CSCI544_Final_Project

1. Project Overview

The goal of our project is performing Aspect Sentiment Analysis on IMDB movie reviews. (Liu, 2015, 5 Aspect Sentiment Classification) Aspect Sentiment Analysis differs from the traditional Sentiment Analysis in that it breaks down review contents into a number of sub components (which is termed “aspects”) and their associated sentiment expression. The Sentiment Index of each aspect is then summarized and reported. At first, we planned to divide aspects in to 6 categories based on the work of Thet et al. (2010). Table 1.1 shows the detail of each aspect:

Aspect	Aspect Description	Aspect keyword
Overall	General feature of a film	MOVIE_GENERAL
Cast	Actor, actor’s performance, casting, etc.	MOVIE_CAST
Director	Director, editing, cinematography, etc.	MOVIE_DIRECTOR
Story	Plot and storyline	MOVIE_STORY
Scene	Scenery, animation, special effect	MOVIE_SCENE
Music	Audio, sound editing, and music	MOVIE_MUSIC

Table 1.1

For each movie, we calculate Sentiment Index for each aspect using lexical-sentiment mapping resources ([7], [8], [9]). The benefit of this approach is that we can provide finer details regarding the movie reviews and the sentiment expression can be used to describe movie's specific characteristics. Noted that we later dropped MOVIE_GENERAL from our consideration as the developer of the Entity Resolution module found that a resolution for this particular aspect is quite challenging and she did not have enough time to properly work on it. Nevertheless, I also include MOVIE_GENERAL as part of the evaluation of my module, as will be described later on in this paper.

We also develop movie review prediction systems. Based on all of the collected reviews associated with a particular movie, we predict whether it received a positive or negative consensus feedback. We consider 2 approaches:

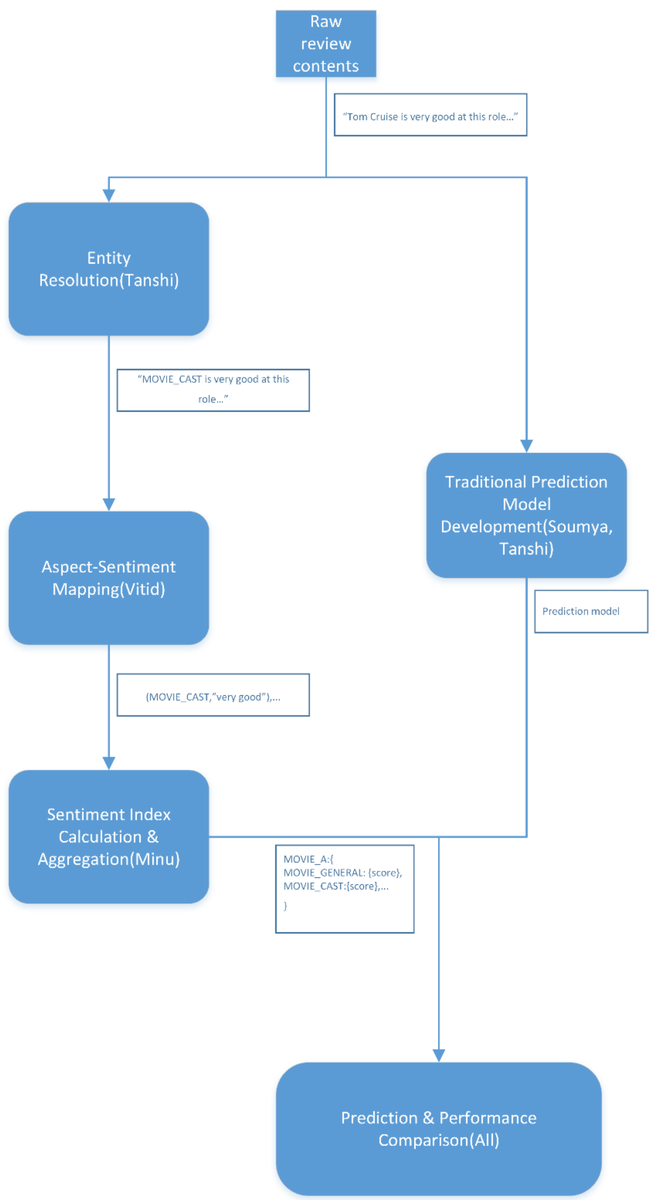
1. Aspect-Sentiment Analysis Approach: Sentiment indexes from all aspects (retrieved from all review contents) are aggregated and used to determine the movie's consensus feedback.
2. Traditional Approach: develop a classification model based on the whole review contents. For each review contents, we extract features, such as bag of words, etc., to predict whether the reviewer gives a positive or negative response. To predict the consensus feedback of a particular movie, we take the majority of the predicted results associated with that film. The dataset used for training the models are obtained from the work of Andrew et al. (2011) [6].

We then compare the prediction performance of both model. The evaluation will be based on Accuracy. We anticipate that Aspect Sentiment Analysis should be helpful in prediction and it will reflect the quality of works done in the pipeline. Additionally, we also hand-annotated a small set of review data to evaluate the Aspect-Sentiment Linking module.

Three members (Tanshi, me, and Minu) are responsible for developing a pipeline process for Aspect Sentiment Analysis. Soumya, and also Tanshi, are responsible for developing a movie review prediction model based on the traditional approach. Our responsibility is summarized in the following table and diagram:

Member	Module	Responsibility
Tanshi Sharma	Aspect Sentiment Analysis	Performing Entity Resolution task: transform a word or word phrase associated with each aspect into Aspect keyword (see: Table 1.1)

Vitid Nakareseisoon		Linking each aspect with its sentiment expression. For example: "The MOVIE_GENERAL is very good" is turned in to (MOVIE_GENERAL, "very good")
Minu George		Calculating Sentiment Index of each sentiment expression and aggregating Sentiment scores for each aspects
Soumy Ravi, Tanshi Sharma	Traditional Movie Review Prediction Model	Developing a movie review prediction model based on the traditional approach (using Machine Learning: Naïve Bayes, etc.)
All	Predict & Performance Comparison	Predicting and Comparing the performance from both approaches

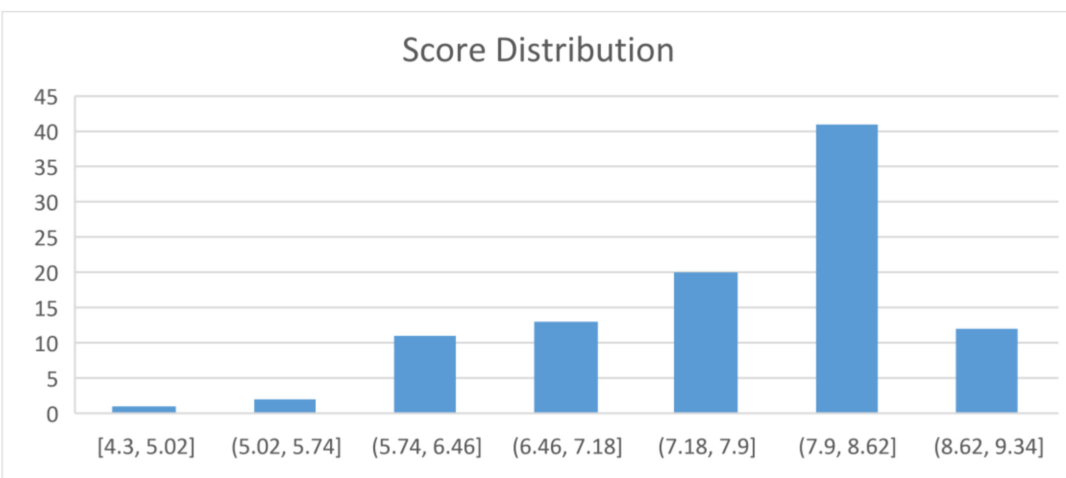


For dataset used in Aspect Sentiment Analysis and the final evaluation part, we crawled review data from IMDB website [1]. The data consisted of ~100 reviews from 100 films, total in around 10,000 review contents. We select the series of top most helpful reviews (indicated as ‘Best’ in IMDB), hoping to reflect the true population’s opinion. A combination of movies that received good and bad scores are selected, as shown in the following table:

Title Id	Name	Review Score
tt0169547	American Beauty	8.4
tt0120586	American History X	8.6
tt2179136	American Sniper	7.3
tt2543164	Arrival	8.4
tt0499549	Avatar	7.9
tt2395427	Avengers: Age of Ultron	7.5
tt0372784	Batman Begins	8.3
tt2245084	Big Hero 6	7.9
tt2562232	Birdman or (The Unexpected Virtue of Ignorance)	7.8
tt1355683	Black Mass	6.9
tt1065073	Boyhood	7.9
tt0112573	Braveheart	8.4
tt3682448	Bridge of Spies	7.6
tt1843866	Captain America: The Winter Soldier	7.8
tt1489889	Central Intelligence	6.4
tt1823672	Chappie	6.9
tt3076658	Creed	7.7
tt2103281	Dawn of the Planet of the Apes	7.6
tt1431045	Deadpool	8.1
tt1860213	Dirty Grandpa	6
tt1840309	Divergent	6.7
tt1853728	Django Unchained	8.4
tt4160708	Don't Breathe	7.3
tt1631867	Edge of Tomorrow	7.9
tt2719848	Everest	7.1
tt0470752	Ex Machina	7.7
tt3183660	Fantastic Beasts and Where to Find Them	7.9
tt1502712	Fantastic Four	4.3
tt2381941	Focus	6.6
tt2820852	Furious 7	7.2
tt1289401	Ghostbusters	5.4
tt0172495	Gladiator	8.5
tt0831387	Godzilla	6.5

tt2267998	Gone Girl	8.1
tt1646971	How to Train Your Dragon 2	7.9
tt1375666	Inception	8.8
tt0361748	Inglourious Basterds	8.3
tt2096673	Inside Out	8.2
tt2908446	Insurgent	6.3
tt0816692	Interstellar	8.6
tt2911666	John Wick	7.2
tt1617661	Jupiter Ascending	5.4
tt2872732	Lucy	6.4
tt1587310	Maleficent	7
tt4046784	Maze Runner: The Scorch Trials	6.4
tt2674426	Me Before You	7.5
tt0209144	Memento	8.5
tt2381249	Mission: Impossible - Rogue Nation	7.4
tt2004420	Neighbors	6.4
tt1959490	Noah	5.8
tt2024469	Non-Stop	7
tt0325980	Pirates of the Caribbean: The Curse of the Black Pearl	8
tt0110912	Pulp Fiction	8.9
tt1234721	RoboCop	6.2
tt3170832	Room	8.2
tt2126355	San Andreas	6.1
tt1700841	Sausage Party	6.4
tt0120815	Saving Private Ryan	8.6
tt0108052	Schindler's List	8.9
tt1130884	Shutter Island	8.1
tt2379713	Spectre	6.8
tt1895587	Spotlight	8.1
tt0076759	Star Wars: Episode IV - A New Hope	8.7
tt0080684	Star Wars: Episode V - The Empire Strikes Back	8.8
tt2488496	Star Wars: The Force Awakens	8.2
tt0103064	Terminator 2: Judgment Day	8.2
tt1872181	The Amazing Spider-Man 2	6.7
tt0848228	The Avengers	8.1
tt1596363	The Big Short	7.8
tt1345836	The Dark Knight Rises	8.5
tt0407887	The Departed	8.5
tt0455944	The Equalizer	7.2
tt2582846	The Fault in Our Stars	7.8

tt0068646	The Godfather	9.2
tt0071562	The Godfather: Part II	9
tt2278388	The Grand Budapest Hotel	8.1
tt0120689	The Green Mile	8.5
tt1951265	The Hunger Games: Mockingjay - Part 1	6.7
tt1951266	The Hunger Games: Mockingjay - Part 2	6.6
tt2084970	The Imitation Game	8.1
tt3040964	The Jungle Book	7.6
tt0918940	The Legend of Tarzan	6.4
tt1490017	The Lego Movie	7.8
tt0167260	The Lord of the Rings: The Return of the King	8.9
tt0167261	The Lord of the Rings: The Two Towers	8.7
tt3659388	The Martian	8
tt0133093	The Matrix	8.7
tt0482571	The Prestige	8.5
tt1663202	The Revenant	8
tt0111161	The Shawshank Redemption	9.3
tt0102926	The Silence of the Lambs	9
tt2980516	The Theory of Everything	7.7
tt0114814	The Usual Suspects	8.6
tt0993846	The Wolf of Wall Street	8.2
tt0120338	Titanic	7.7
tt1964418	Tomorrowland	6.5
tt0434409	V for Vendetta	8.2
tt2582802	Whiplash	8.5
tt1877832	X-Men: Days of Future Past	8
tt2948356	Zootopia	8.1



<u>Classification Result</u>

We consider movies that received a rating of 7 or higher as a positive class. The accuracy of all approaches are as follow:

	Aspect Sentiment Analysis	Traditional Movie Review Prediction Model						
		Linear SVM	RBF SVM	Logistics Regression	AdaBoosts	Naïve Bayes	Max Entropy	Random Forest
Accuracy (%)	66	72	69	**74**	70	70	67	65

Additionally, we also collected Precision, Recall, and F1 Score of Aspect Sentiment Analysis approach:

True consensus sentiment	Precision	Recall	F1 Score
Positive	78 %	75 %	76 %
Negative	42 %	46 %	44 %

Base on the result, we found that simple prediction models, such as Logistics Regression, Linear SVM, and Naïve Bayes, are very effective in predicting the movie's consensus feedback. This may suggest a linear relationship of the extracted features, or the overfitting problem of the non-linear models. The prediction based on Aspect Sentiment Analysis approach does not perform as well as we expected. This can be based on various errors that propagate throughout our pipeline process, such as aspects are not resolved by the Entity Resolution module, sentiment expressions are not correctly linked by the Aspect-Sentiment Mapping module, or Sentiment Indexes are not properly calculated by the Sentiment Index Calculation module. Because of the limited timeframe, we do not have enough time to improve our processes further. However, Aspect Sentiment Analysis still shows a very promising way to be used for sentiment classification, as it performs relatively well against the state-of-the-art Random Forest model.

From the second table, it is obvious that the performance of Aspect Sentiment Analysis approach is hampered by the movies that received a negative consensus feedback. The problem associated with negative comments is also pointed out later on in my module's evaluation. Based on our observation, irony style frequently used in the negative comments can confuse the Aspect-Sentiment Mapping module. The aspect may be linked with an expression that has the opposite sentiment direction (from what it should be). Such errors can be accumulated and propagated downward the pipeline, resulted in the noticeably lower Precision, Recall, and F1 Score for the negative class.

2. My Primary Responsibility

Goal

My main responsibility is developing a module that links a given aspect token with its associated sentiment expression. My module receives an input file (in Json form) from Entity Resolution module and output a CSV file consisted of tuple: (index, aspect, sentiment_expression, conjunction, conjunction_expression). For example, if the input file is:

```
[
{“review”: “MOVIE_CAST is great. MOVIE_GENERAL is very good…”},
{“review”: “A MOVIE_CAST is not good but not bad…”},…
]
```

My module will produce the following CSV output:

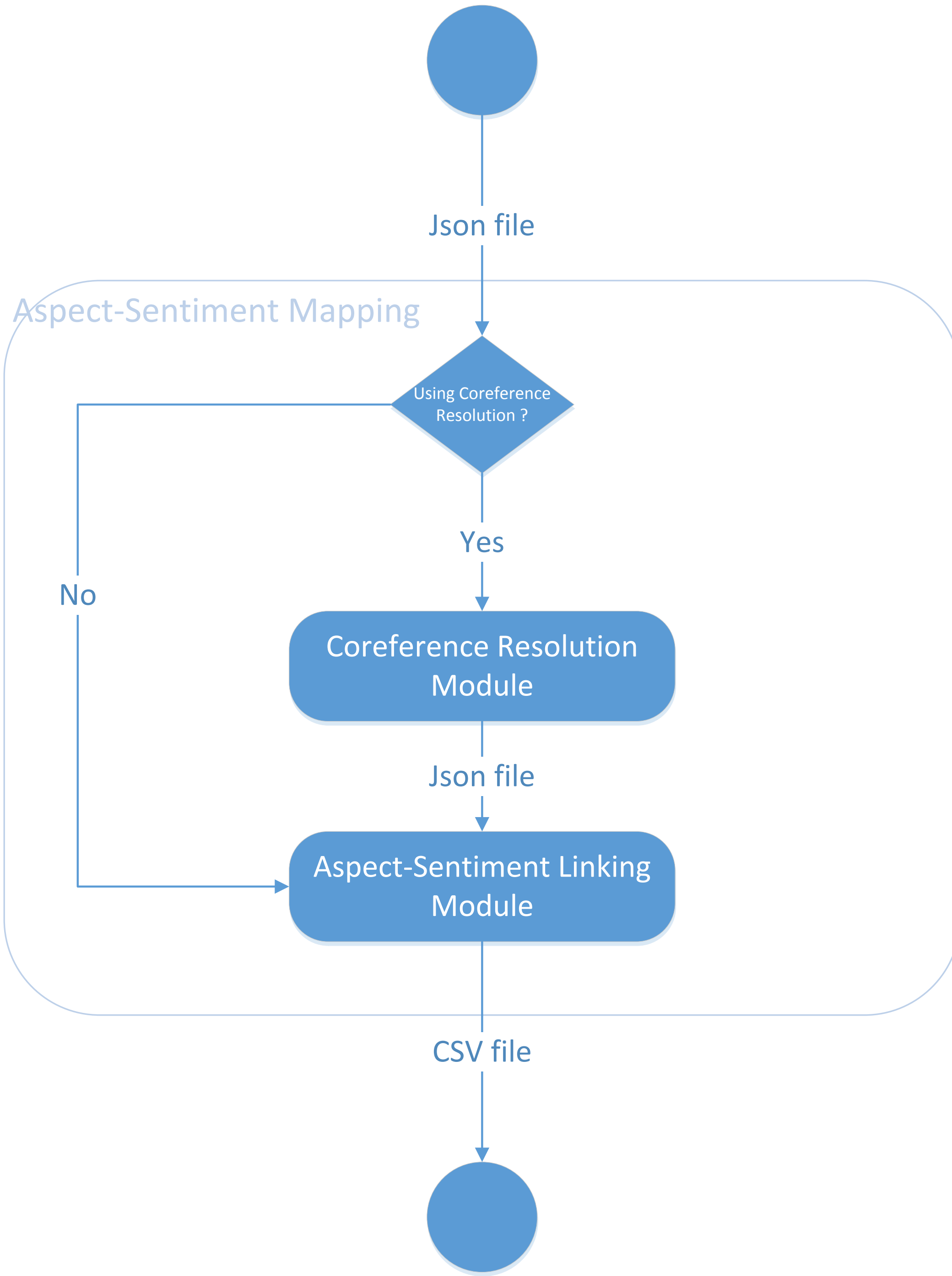
```
index,aspect,sentiment_expression,con,conjSentiment
1, MOVIE_CAST, great/JJ,,
1, MOVIE_GENERAL, very/RB good/JJ,,
2, MOVIE_CAST, not/RB good/JJ, but/CC, not/RB, bad/JJ
2, MOVIE_CAST, not/RB bad/JJ, but/CC, not/RB, good/JJ
…
```

Noted that the result is POS-tagged and linked with conjunction sentiment (if any) that will be used by the downstream process in the pipeline.

I use Stanford CoreNLP suite (version 3.7.0-beta) [2] as an NLP engine in my module development. I developed my program in Java language and included several other 3rd party libraries. The program is completely standalone from other processes in the pipeline and can be run separately.

Methodology

My module consisted of 2 parts: Coreference Resolution Module and Aspect-Sentiment Linking Module, as shown in the below diagram:

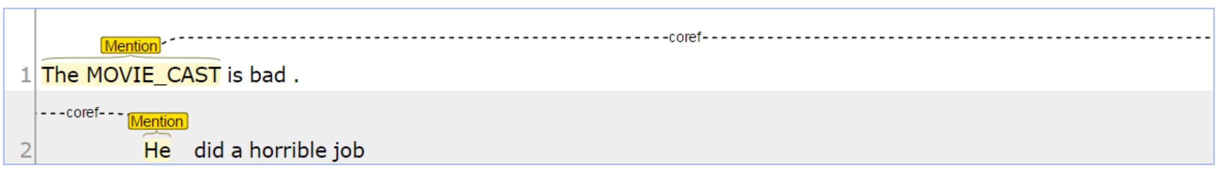


Coreference Resolution Module

From the given input file, the module performs Coreference Resolution, using multi-pass sieve anaphora resolution (Lee et al., 2011), on the review content and replace referred tokens with their represented token. For example, given the following input:

The MOVIE_CAST is bad. He did a horrible job…

The following coreference chain can be detected by Anaphora Resolution:



The module will use the obtained coreference chain to replace “He” with “The MOVIE_CAST”:

MOVIE_CAST is bad. The MOVIE_CAST did a horrible job…

After resolving all coreference chains in a review content, the result will be collected in an intermediate Json file. This module is designed to be completely optional. Performing Reference Resolution is computationally intensive and can be skipped entirely (Noted: we applied coreference resolution to all reviews used in the final evaluation).

Aspect-Sentiment Linking Module

The module reads the intermediate output, performs Dependency Parsing (Marneffe & Manning, 2008), and maps a potential sentiment expression with the aspect token keywords using the obtained dependency graph. The mapping is based the predefined syntactic rules (Liu, 2015, 6.2 Exploiting Syntactic Relations) and configurable. The grammar of the rules is based on Semgrex pattern [3].

The configuration file is in:

https://github.com/vitid/SentimentExtractor/blob/master/src/main/resources/extract_rules.test.properties

The final mapping rules used are:

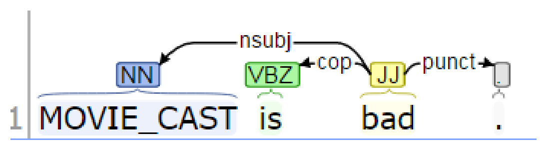
```
rule.adj.0={tag:/JJ.*/} >/nsubj.*/ {word:_ASPECTS_;tag:/NN.*/}
rule.adj.1={tag:/JJ.*/} >/nsubj.*/ ( {} >/nmod.*/ {word:_ASPECTS_;tag:/NN.*/})
rule.adj.2={tag:/JJ.*/} </amod.*/ {word:_ASPECTS_;tag:/NN.*/}
rule.compound_noun.0={tag:/NN.*/} </compound.*/ {word:_ASPECTS_;tag:/NN.*/}
rule.noun.0={tag:/NN.*/} >/nsubj.*/ {word:_ASPECTS_;tag:/NN.*/}
rule.noun.1={tag:/NN.*/} </nsubj.*/ {word:_ASPECTS_;tag:/NN.*/}
rule.noun.2={tag:/NN.*/} >/nmod.*/ {word:_ASPECTS_;tag:/NN.*/}
rule.subject_of_verb.0={tag:/VB.*/} >/nsubj.*/ {word:_ASPECTS_;tag:/NN.*/}
```

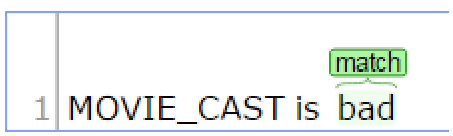
(Noted, the token word _ASPECTS_ will be turned to one of the predefined aspect token keywords).

The detail of each rules is as follow:

- rule.adj.0={tag:/JJ.*/} >/nsubj.*/ {word:_ASPECTS_;tag:/NN.*/}

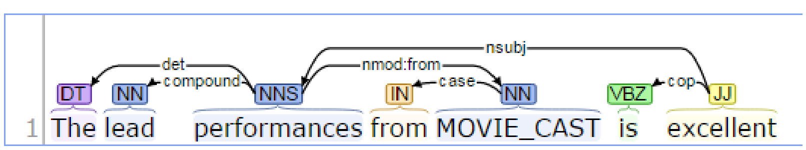
  Map adjective with an aspect when the aspect acts as the subject of the modifier clause. For example,

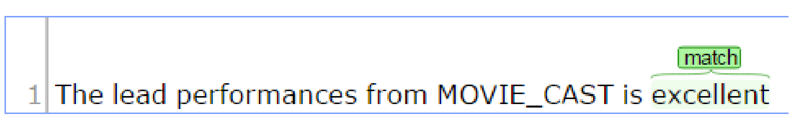




- rule.adj.1={tag:/JJ.*/} >/nsubj.*/ ({} >/nmod.*/ {word:_ASPECTS_;tag:/NN.*/})

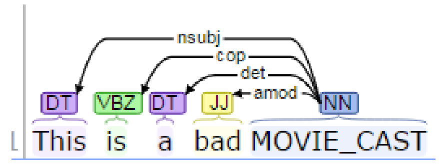
  Similar to the above case, with an indirect reference to aspect, For example,

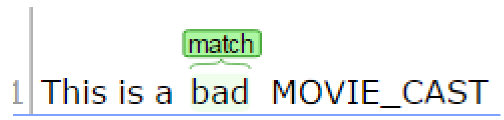




- rule.adj.2={tag:/JJ.*/} </amod.*/ {word:_ASPECTS_;tag:/NN.*/}

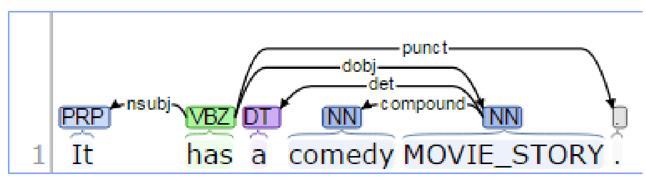
  Map adjective with an aspect when the adjective is the linked by a modifier relation. For example,

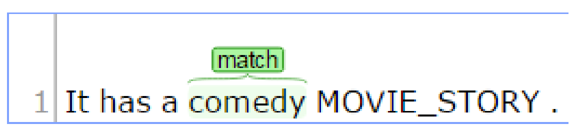




- rule.compound_noun.0={tag:/NN.*/} </compound.*/ {word:_ASPECTS_;tag:/NN.*/}

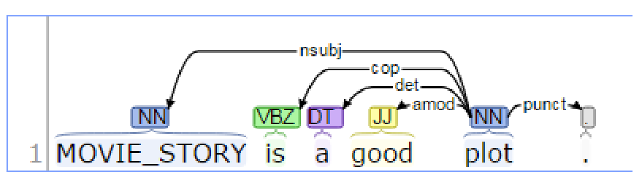
  Map an aspect with its compound noun. For example,

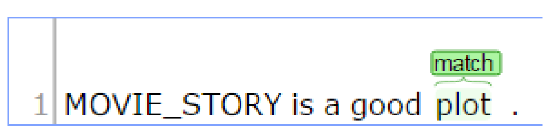




- rule.noun.0={tag:/NN.*/} >/nsubj.*/ {word:_ASPECTS_;tag:/NN.*/}

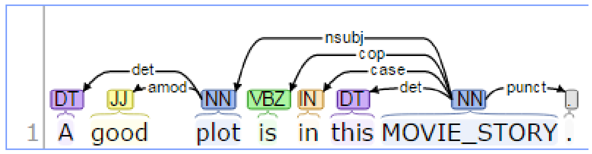
  Map an aspect with a noun that describes it. For example,

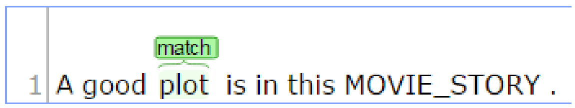




- rule.noun.1={tag:/NN.*/} </nsubj.*/ {word:_ASPECTS_;tag:/NN.*/}

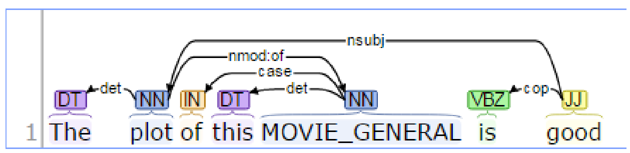
  Similar to the above, with a reverse direction. For example,

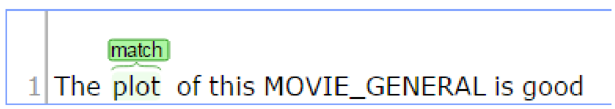




- rule.noun.2={tag:/NN.*/} >/nmod.*/ {word:_ASPECTS_;tag:/NN.*/}

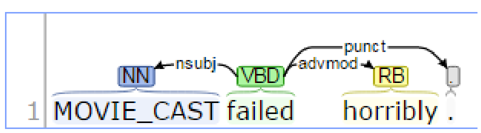
  Similar to the above, with a modifier relation. For example,

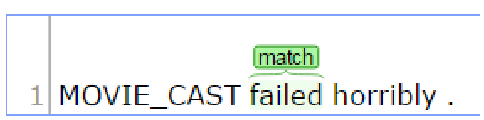




- rule.subject_of_verb.0={tag:/VB.*/} >/nsubj.*/ {word:_ASPECTS_;tag:/NN.*/}

  Map an aspect with a verb when the aspect acts as a subject of the verb phrase. For example,





In summary, the main relationships that the module captured are:

Relationship	Example Sentence
ASPECT <-> Adjective	MOVIE_GENERAL is good, It has a good MOVIE_MUSIC

ASPECT<->Noun	MOVIE_GENERAL is the scariest film I have ever seen
Verb phrase with Aspect is a subject	MOVIE_CAST did a horrible job, MOVIE_STORY fails

Noted that since the goal of this module is also capturing only attributes that are useful in describing an aspect, the relationship where an aspect acts as an object is ignored. Consider the following sentence:

I love MOVIE_GENERAL

Here, “love” describes the reviewer, not the aspect, and is excluded from the rules.

After the root sentiment is retrieved, the module will try to expand the expression based on the internal rules. For example, consider the root sentiment “good” captured from:

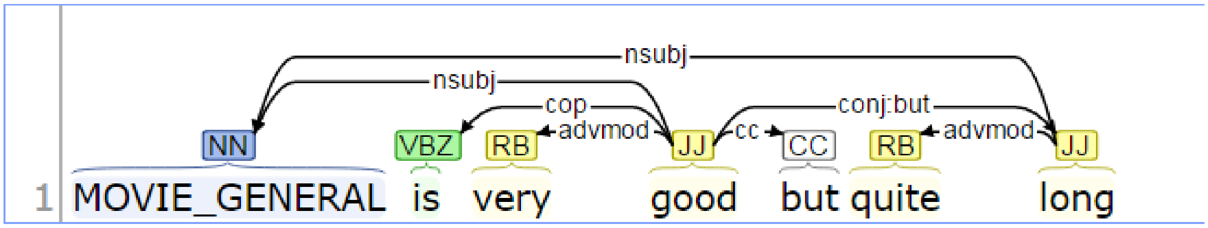
MOVIE_GENERAL is very good

From the example above, Adjective (POS tag: JJ) will be extended by Adverb (POS tag: RB). The expand rules are limited so that tokens that are not associated with the aspect should not be included. So, from the example above, MOVIE_GENERAL is linked with “very good”. It then looks for any possible conjunction clauses and associated conjunction sentiment. The result is then POS-tagged, collected, and written to a CSV file. So, the output of this sentence will be:

```
index,aspect,sentiment,conj,conjSentiment
1, MOVIE_GENERAL, very/RB good/JJ,,
…
```

For more complicate example, if the sentence is:

MOVIE_GENERAL is very good but quite long



The output of the sentence will be

```
index,aspect,sentiment,conj,conjSentiment
1, MOVIE_GENERAL, very/RB good/JJ, but/CC, quite/RB long/JJ
1, MOVIE_GENERAL, quite/RB long/JJ, but/CC, very/RB good/JJ
…
```

Noted: the index column refers to the review index in the input file (mainly used for a module evaluation and Error Analysis). The reason that the output is paired with a conjunction clause is because the Sentiment Index Calculation module (in the downstream process) can use a conjunction clause to infer the sentiment score when the sentiment score from the primary sentiment expression cannot be inferred (Thet et al., 2010). For example, the primary expression "quite long" does not carry a negative connotation, but the positive sentiment can be calculated from "very good" and then reversed with "but" conjunction to derive that "quite long" carries a negative connotation.

A considerable amount of improvement based on Error Analysis had been done. At first, I used simple syntactic rules to capture Aspect-Sentiment relations. It turned out that many irrelevant information was also collected. In another case, a critical piece of information such as a pre-conjunction "neither…nor", which can shift a sentiment score was not collected. I used log files printed out from the module to investigate common types of expression used, track down what can possibly go wrong and incrementally updated the syntactic rules.

<u>Module Evaluation</u>

To evaluate the Aspect-Sentiment Linking module, I hand-annotated a small set of reviews from 3 movies. Because of the highly subjective nature of Sentiment linking, I evaluated my model based solely on the Precision metric. The conjunction clause is ignored and the relation retrieved is considered "correct" if a critical piece of sentiment is gathered and the sentiment polar is not changed. For example, given the following sentence:

The MOVIE_GENERAL is not that good

The following retrieved tuples are valid:

(MOVIE_GENERAL, not that good)
(MOVIE_GENERAL, not good)

While the following tuples are not valid:

(MOVIE_GENERAL, good)
(MOVIE_GENERAL, is)
(MOVIE_GENERAL, is not)

Furthermore, a duplicated sentiment expression will also be considered invalid.

The following table displays the module's evaluation result for each movies

Movie	#reviews	#relations captured	Precision (%)
Armageddon (1998)	36	178	55 %
The cabin in the Woods (2012)	26	91	45 %
Into the woods (2014)	19	151	37 %

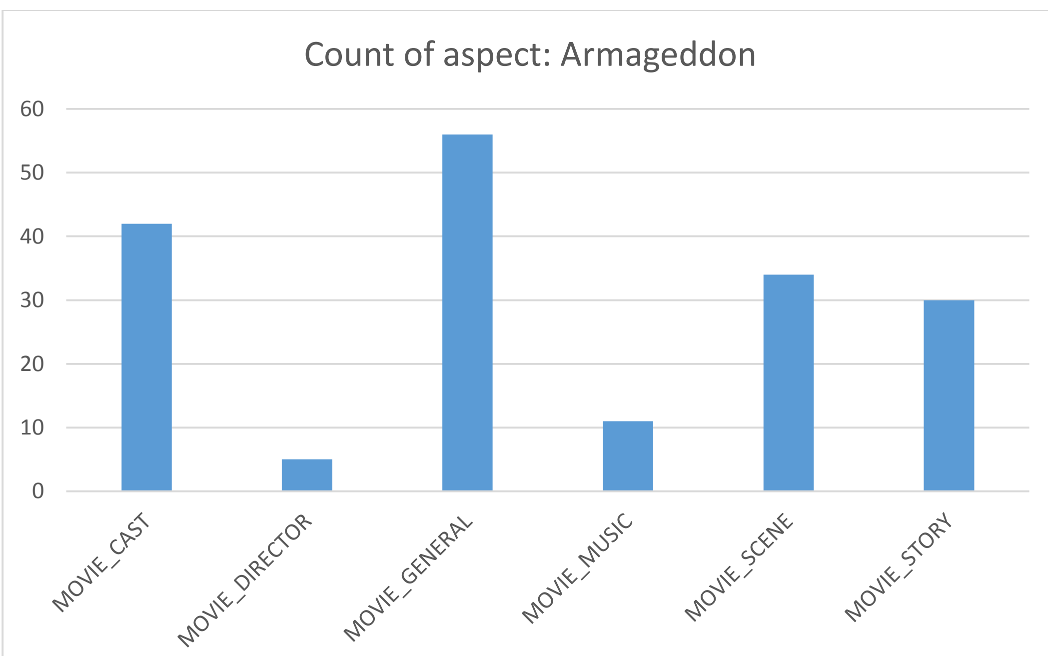
The following table displays the result based on all 3 movies, and also aggregated by the aspect type

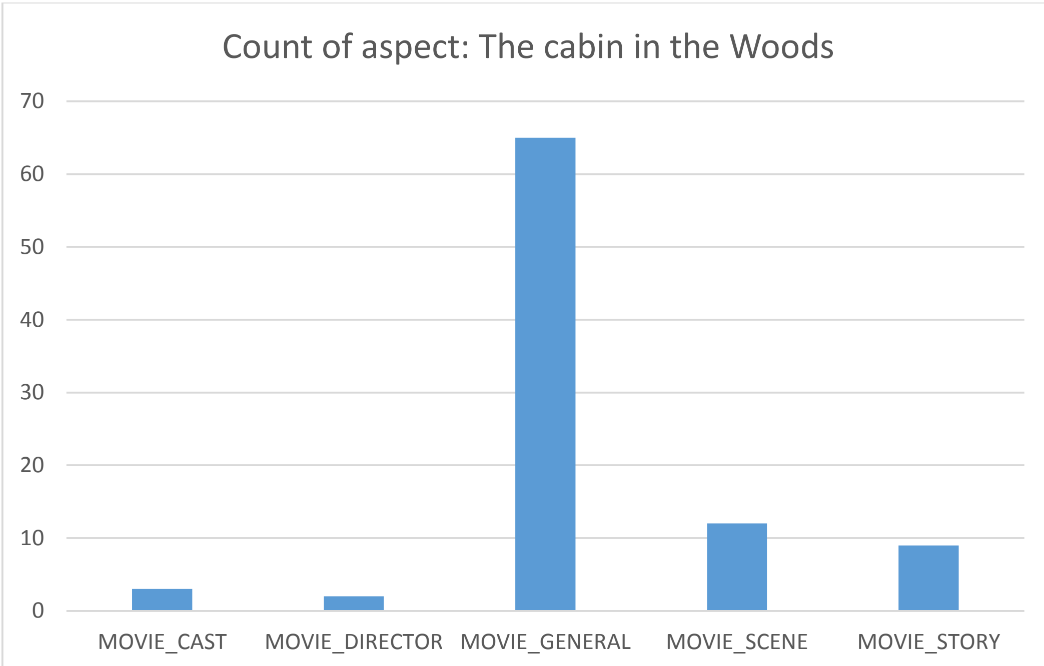
Type	Precision (%)
All Aspects	46 %
MOVIE_GENERAL	48 %
MOVIE_CAST	36 %
MOVIE_DIRECTOR	57 %

MOVIE_STORY	47 %
MOVIE_SCENE	53 %
MOVIE_MUSIC	51 %

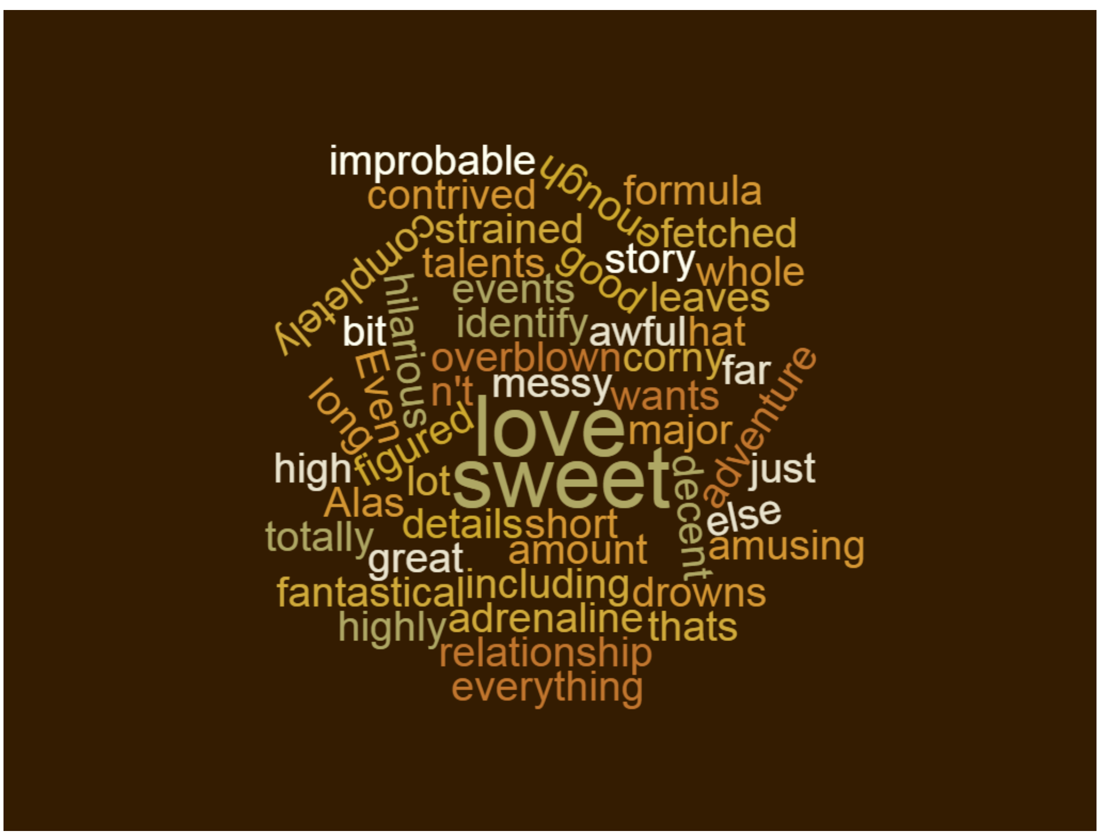
Based on the result, Aspect-Sentiment Linking is a fairly hard problem. The precisions of the module barely go above 50%. Upon a closer inspection, I found that the performance tends to get worse for a movie that received a lot of negative comments (as in the case of the latter 2 films); satire tend to be used in a negative comment and can confuse the mapping. The bottom line is that mapping expression maybe less accurate in the case of negative review. The upside is that many irrelevant expressions that are captured, such as “are”, “were”, and “whole”, do not hold sentiment score and should not have much effect on the final calculated sentiment score.

Moreover, the relationship retrieved can be used to infer other information from the movie. For example, the following histogram charts show the count of aspects captured from Sentiment-linking. For Armageddon, people tend to talk about MOVIE_CAST, as the movie comprised of many well-known actors/actresses. While in The cabin in the Woods, it comprised of lesser-known casts and people tend to direct their opinion toward the overall general aspect of the movie.





Sentiment expressions can also be used to analyze what people tend to talk about regarding the associated aspect. The following picture shows the word cloud [4] of Sentiment expression of Armageddon's MOVIE_STORY aspect:



The associated artifacts; evaluation result and input & output files of the module, as well as application logs can be found in:
https://github.com/vitid/CSCI544_Final_Project/blob/master/project_artifacts.zip .

3. Other Project Work

I developed IMDB reviews crawler module: "ScrapyIMDB" to crawl a list of movies and review contents from IMDB. It is written in Python and uses Scrapy [5] as a crawler engine. I decided what movies to crawl and performed the data collection itself; crawled a list of 100 movies, and all 10,000 review contents (see: Table 1.2). I also crawled and hand-annotated a smaller set of movie reviews to evaluate the module that I developed (see section 2.). Lastly, I prepared the presentation slides concerning my module (Aspect-Sentiment Mapping).

4. Online Resources

1. IMDB. Our review contents are collected from the website. http://www.imdb.com/
2. Stanford CoreNLP, NLP engine used in my module. http://stanfordnlp.github.io/CoreNLP/
3. Semgrex, Grammar pattern for matching Stanford CoreNLP's Dependency Graph.
   http://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/semgraph/semgrex/SemgrexPattern.html
4. Word Cloud Generator, Website used for generating the word cloud picture.
   http://www.wordclouds.com/
5. Scrapy, Crawling library for Python. https://scrapy.org/
6. Stanford's Large Movie Review Dataset. Dataset used for developing the prediction models.
   http://ai.stanford.edu/~amaas/data/sentiment/
7. SentiWordNet. Lexical resource for Sentiment calculation. http://sentiwordnet.isti.cnr.it/
8. Vader Sentiment Analyzer. Lexical resource for Sentiment calculation.
   https://github.com/cjhutto/vaderSentiment
9. AFINN. Lexical resource for Sentiment calculation.
   http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010

5. References

Liu, B., 1963. (2015). 5 Aspect Sentiment Classification, *Sentiment analysis: Mining opinions, sentiments, and emotions*. New York: Cambridge University Press.

Liu, B., 1963. (2015). 6.2 Exploiting Syntactic Relations, *Sentiment analysis: Mining opinions, sentiments, and emotions*. New York: Cambridge University Press.

Marneffe, M. D., & Manning, C. D. (2008). The Stanford typed dependencies representation. Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation - CrossParser '08. doi:10.3115/1608858.1608859

Thet, T. T., Na, J., & Khoo, C. S. G. (2010). Aspect-based sentiment analysis of movie reviews on discussion boards. Journal of Information Science, 36(6), 823-848. doi:10.1177/0165551510388123

Lee, H., Peirsman, Y., Chang, A., Chambers, N., Surdeanu, M. & Jurafsky, D. (2011). Stanford's Multi-Pass Sieve Coreference Resolution System at the CoNLL-2011 Shared Task. In Proceedings of the CoNLL-2011 Shared Task, 2011.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).