ASSESSMENT COVER SHEET

Assignment Details

Course: Applied Natural Language Processing

Semester/Academic Year: Semester 1/ 2022

Assignment title: Building an aspect-based sentiment analysis algorithm based on syntactic parsing

Assessment Criteria

Assessment Criteria are included in the Assignment Descriptions that are published on each course's web site.

Plagiarism and Collusion

Plagiarism: using another person's ideas, designs, words or works without appropriate acknowledgement.

Collusion: another person assisting in the production of an assessment submission without the express requirement, or consent or knowledge of the assessor.

Consequences of Plagiarism and Collusion

The penalties associated with plagiarism and collusion are designed to impose sanctions on offenders that reflect the seriousness of the University's commitment to academic integrity. Penalties may include: the requirement to revise and resubmit assessment work, receiving a result of zero for the assessment work, failing the course, expulsion and/or receiving a financial penalty.

DECLARATION

I declare that all material in this assessment is my own work except where there is clear acknowledgement and reference to the work of others. I have read the University Policy Statement on Plagiarism, Collusion and Related Forms of Cheating:

http://www.adelaide.edu.au/policies/?230

I give permission for my assessment work to be reproduced and submitted to academic staff for the purposes of assessment and to be copied, submitted and retained in a form suitable for electronic checking of plagiarism.

13-6-2022

SIGNATURE AND DATE

Applied Natural Language Processing (COMP SCI 7417)
Assignment 3 Report

Cheuk Hang Ng (a1821087)

University of Adelaide, a1821087@student.adelaide.edu.au

1 INTRODUCTION

This assignment aims to write a code in Python for building an aspect-based sentiment analysis algorithm based on syntactic parsing. The code was written and run on Jupyter Notebook Python environment. The dataset is an XML file provided by the course, which consists of data such as “sentence_id”, “text”, “aspectTerm”, “aspectCategories1” and “polarity”. In this assignment, we only considered the data contained in “text”, “aspectTerm” and “polarity”. “text” contains sentences that to be used for sentiment analysis while “aspectTerm” and “polarity” contains terms to be looked at and the ground truths of the sentiment of the terms respectively. The dataset was first read as a dictionary and investigated. Some manipulations were done before used for dependency parsing. For example, the dictionary was modified as a more readable and user-friendly sturcture, and the aspect terms and texts were pre-processed. All texts underwent dependency parsing and the sentiment of each aspect term in the texts were analyzed. For each polarity, namely “positive”, “negative and “neutral”, we used three rules to determine the sentiment of the aspect term. The performances of each rule were evaluated by their precisions and recalls.

2 DESCRIPTIONS OF DETAILS

2.1 Data and Text Pre-processing

The data provided for this assignment was in XML format, which cannot be directly read and therefore, the data had to be pre-processed before moving to the next step. In this assignment, we used the xmltodict and json library to assist in reading the XML file as a dictionary object, snippets of the dictionary before and after modified were shown below.

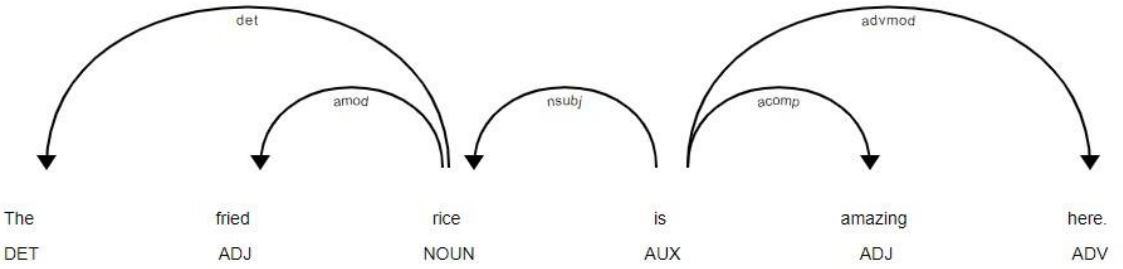
```
In [40]: with open('Restaurants.xml', 'r', encoding='utf-8') as file:
             my_xml = file.read()

         my_dict = xmltodict.parse(my_xml)

         pprint.pprint(my_dict)

         OrderedDict([('sentences',
                       OrderedDict([('sentence',
                                     [OrderedDict([('@id', '3121'),
                                                   ('text',
                                                    'But the staff was so horrible to '
                                                    'us.'),
                                                   ('aspectTerms',
                                                    OrderedDict([('aspectTerm',
                                                                  OrderedDict([('@term',
                                                                                'staff'),
                                                                               ('@polarity',
                                                                                'negative'),
                                                                               ('@from',
                                                                                '8'),
                                                                               ('@to',
                                                                                '13')]))])),
                                                   ('aspectCategories',
                                                    OrderedDict([('aspectCategory',
                                                                  OrderedDict([('@category',
```

```
In [12]: original_all_sentences = all_sentences

         original_all_sentences

Out[12]: {0: {'id': '3121',
               'sentence': 'But the staff was so horrible to us.',
               'aspectTerm': ['staff'],
               'polarity': ['negative'],
               'span': ('8', '13')},
          1: {'id': '2777',
              'sentence': "To be completely fair, the only redeeming factor was the food, which was above average, but couldn't make up f
         or all the other deficiencies of Teodora.",
              'aspectTerm': ['food'],
              'polarity': ['positive'],
              'span': ('57', '61')},
          2: {'id': '1634',
              'sentence': "The food is uniformly exceptional, with a very capable kitchen which will proudly whip up whatever you feel li
         ke eating, whether it's on the menu or not.",
              'aspectTerm': ['food', 'kitchen', 'menu'],
              'polarity': ['positive', 'positive', 'neutral'],
              'span': [('4', '8'), ('55', '62'), ('141', '145')]},
          5: {'id': '2846',
              'sentence': "Not only was the food outstanding, but the little 'perks' were great.",
```

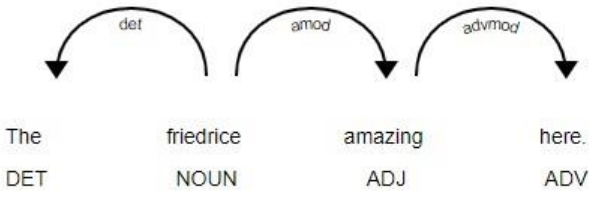
By exploring the modified dictionary, we found that there were aspect terms that consist of more than one word. In term of dependency parsing, this might affect the parsing result and the accuracy of finding the corresponding

sentiment words. Therefore, we modified these aspect terms into one single word despite losing their English meaning. In addition, we removed stopwords in all texts, where the list of stopwords was imported from nltk.corpus library and was slightly modified to exclude the below words: 'but', 'no', 'not', 'too', 'very', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't". The pre-processed texts were examined with visualization of the dependency parsing results with the libraries from spaCy. In short, these pre-processes favored the parsing. Most cases of the multi-words aspect term after merged were identified as noun by the dependency parser which helped in sentiment words searching, and the stopwords removal shortened the relation distances between words.

Token	Relation	Head	Children
The	det	rice	[]
fried	amod	rice	[]
rice	nsubj	is	[The, fried]
is	ROOT	is	[rice, amazing, here, .]
amazing	acomp	is	[]
here	advmod	is	[]
.	punct	is	[]



Token	Relation	Head	Children
The	det	friedrice	[]
friedrice	ROOT	friedrice	[The, amazing, .]
amazing	amod	friedrice	[here]
here	advmod	amazing	[]
.	punct	friedrice	[]



2.2 Aspect-Based Sentiment Analysis

To perform aspect-based sentiment analysis, all sentences were parsed by the dependency parser from the spaCy library and attempted to search for the corresponding adjectives of the aspect terms by looking at its children, its ancestors, the children of its ancestors and the children of its children one by one. It is believed that these positions covered most of the possible positions of the adjectives for a specific word in an English sentence. The words found in these positions were put into a list if it satisfied one of the conditions of: being tagged as “ADJ” for part-of-speech tagging, or the dependency type is either “acomp”, “amod” or “advmod”. And the first word in the list would be passed to the function TextBlob(text).sentiment/ TextBlob(text).polarity to determine the polarity of the word. A function was written to transform the numeric result from TextBlob function to a string that represents the polarity, where it returns ‘positive’ if the polarity score is greater than 0.1, returns ‘negative’ if the polarity score is smaller than -0.1, and returns ‘neutral’ otherwise.

A list of negation words was also defined at this stage for later use, which contained the following words: "no", "not", "n't", "none", "however", "but", "nevertheless", "nonetheless”, "yet", "neither", "nor", "never", "any", "hardly" and "barely". We believed that these are the most common words used in an English sentence for negation.

For each polarity, three rules were decided for predicting the sentiment of the aspect terms.

2.2.1 Rules for positive sentiment

I. Rule 1 for positive sentiment: The first rule used for predicting positive sentiment was to look at the closet adjective to the aspect term. We defined closet by the dependency relation here, where an immediate child is closer to an immediate ancestor, and immediate children and ancestors are closer than the children of the term’s ancestors and the children of the term’s children.

II. Rule 2 for positive sentiment: Next, we look for ‘negative’ adjectives with negation in front of it. By negating a negative adjective, the sentiment was considered as positive and therefore used as predicting positive sentiment. For each aspect term, if we found any adjective related to it, we would try to look for negation words (i.e. words in previous negation words list) in two or one word before that adjective. Only when we found any, we determine its polarity, otherwise it is considered as ‘neutral’ even if the adjective by itself is ‘negative’. Below is a code snippet for this rule:

```
for k, v in all_sentences.items():
    sentence = v['sentence']
    doc = nlp(sentence)
    for term in v['aspectTerm']:
        ground_truth = v['polarity'][v['aspectTerm'].index(term)]
        if (ground_truth != 'conflict'):
            for token in doc:
                if token.text == term:
                    adj_list = find_adj(token)
                    pred_pol = 'neutral'
                    if len(adj_list) > 0:
                        term_index = [d.text for d in doc].index(term)
                        if term_index - 2 > 0:
                            if [d.text for d in doc][term_index - 2].lower() in negate_list:
                                polarity, subjectivity = TextBlob(adj_list[0]).sentiment
                                if polarity < -0.1:
                                    pred_pol = det_polarity(-polarity)
                            if [d.text for d in doc][term_index - 1].lower() in negate_list:
                                polarity, subjectivity = TextBlob(adj_list[0]).sentiment
                                if polarity < -0.1:
                                    pred_pol = det_polarity(-polarity)
                        elif term_index - 1 > 0:
                            if [d.text for d in doc][term_index - 1].lower() in negate_list:
                                polarity, subjectivity = TextBlob(adj_list[0]).sentiment
                                if polarity < -0.1:
                                    pred_pol = det_polarity(-polarity)
```

And we changed the polarity of the adjective by multiplying the polarity score with -1.

III. Rule 3 for positive sentiment: We consider adjectives where there is an adverb immediately in front of it. In English, an adverb could be used to modify the sentiment of an adjective, and we tried to look for this kind of adjectives, by using similar techniques in Rule 2. We multiply the polarity score by 1.5 before feeding it into the polarity classifying function. Only those adjectives with adverb in front of it were considered, otherwise it was determined as 'neutral'.

2.2.2 Rules for neutral sentiment

I. Rule 1 for neutral sentiment: The first rule was simple. If there was not any adjectives found with the aspect term, or the adjective found was determined as 'neutral', we predicted that the sentiment of the aspect term was 'neutral'. The first half of the rule was due to the reason that in most cases, if no adjective was used with the aspect term, it is usually not expressing any sentiment, while the second half of the rule is straightforward.

II. Rule 2 for neutral sentiment: We again looked for negation of negative adjectives. This rule was set because in English, people sometimes express a neutral feeling by negating some negative descriptions, where it sounds more polite in this way. The logic and techniques used in this rule were exactly the same as Rule 2 for positive sentiment, except we defined an empty string for the predicted polarity instead of a 'neutral' polarity.

III. Rule 3 for neutral sentiment: For the third rule for predicting 'neutral' sentiment, we took the average of the polarity scores of all adjectives found for a specific aspect term. We believed that sometimes people use more than one adjective to describe things if they have an uncertain sentiment. Hence taking an average of the polarity scores of the adjectives for an aspect term would help identifying an neutral sentiment. Below is a code snippet for determining polarity with average polarity scores:

```
for k, v in all_sentences.items():
    sentence = v['sentence']
    doc = nlp(sentence)
    for term in v['aspectTerm']:
        ground_truth = v['polarity'][v['aspectTerm'].index(term)]
        if (ground_truth != 'conflict'):
            for token in doc:
                if token.text == term:
                    adj_list = find_adj(token)
                    pred_pol = ""
                    if len(adj_list) > 0:
                        pol_list = [TextBlob(x).polarity for x in adj_list]
                        pred_pol = det_polarity(mean(pol_list))
```

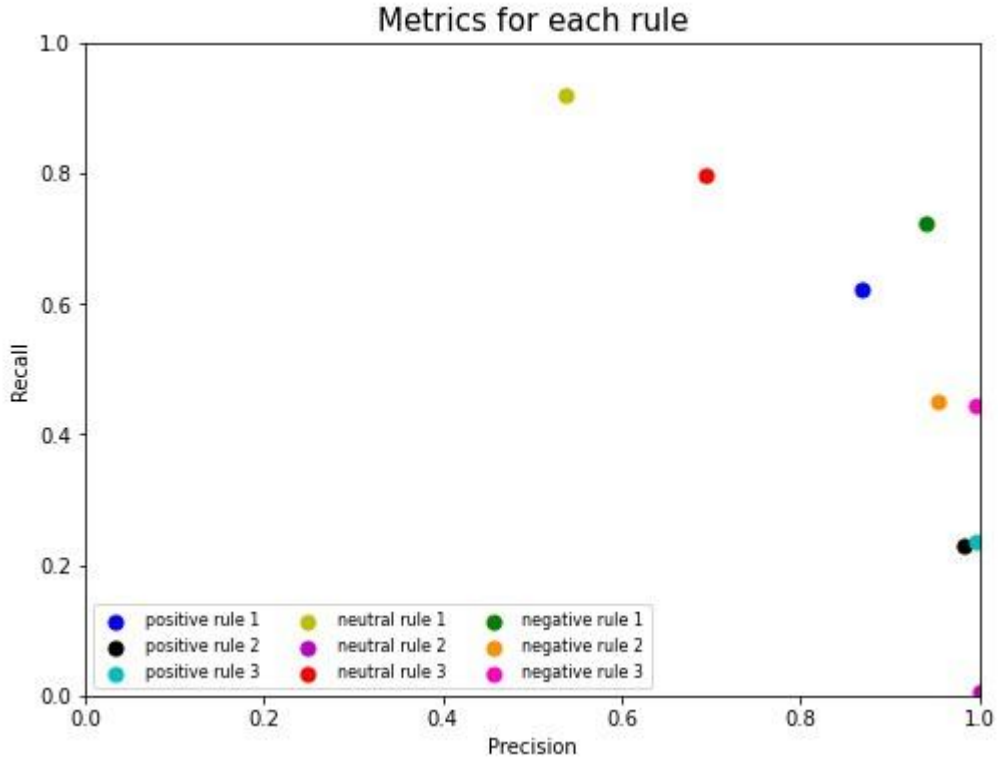
2.2.3 Rules for negative sentiment

The rules for predicting negative sentiment were basically identical to those for predicting positive sentiment, except for the polarity was changed to negative instead of positive. The rules were: I. Looking at the closet adjective to the aspect term; II. Looking for 'positive' adjectives with negation in front of the adjective related to the term; and III. Looking for adjectives related to the aspect term where there is an adverb immediately in front of it.

3 EVALUATION AND RESULTS COMPARISON

Each rule was evaluated with precision and recall. Precision was calculated by the ratio between true positives and the sum of true positives and false positives. Recall was calculated by the ratio between true positives and the sum of true positives and false negatives.

The scores for each rule were visualized as follow:



Notice that Rule 1 for both negative and positive sentiment prediction and Rule 3 for neutral sentiment outperformed the rests in terms of having relatively balanced precisions and recalls, which means that these rules were able to capture the sentiments in general. Note that both Rule 2 and 3 for positive sentiment have lower recall than Rule 2 and 3 for negative sentiment, although logically they are identical and having similar precisions. A low recall means that the rule is unable to identify some positive case. Recall that in Rule 2 and Rule 3 for positive and negative sentiments, we tried to identify the sentiment with negations and adverbs. Rule 2 and 3 for predicting positive sentiment having lower recalls than those for predicting negative sentiment indicating that in the data, the positive sentiments were usually expressed directly without any use of negations and adverbs, which matches my impression in the real world: compliments are usually expressed directly but negative sentiments are sometimes exaggerated with adverb and negations of positive words. On the other hand, Rule 1 for predicting neutral sentiment has a relatively high recall but moderate precision. This could be understood because in real world, people could express their thoughts and feelings obscurely without using any adjectives in a sentence, which results in numbers of false positive cases. A special case is the Rule 2 of predicting neutral sentiment. It has a precision with 1 but almost 0 recall. It captured all neutral sentiment without

false positive case but failed to identify most neutral sentiment statements. It was interpreted as the rule being too narrow for detecting neutral sentiment.