Assignment 4: Natural Language Understanding

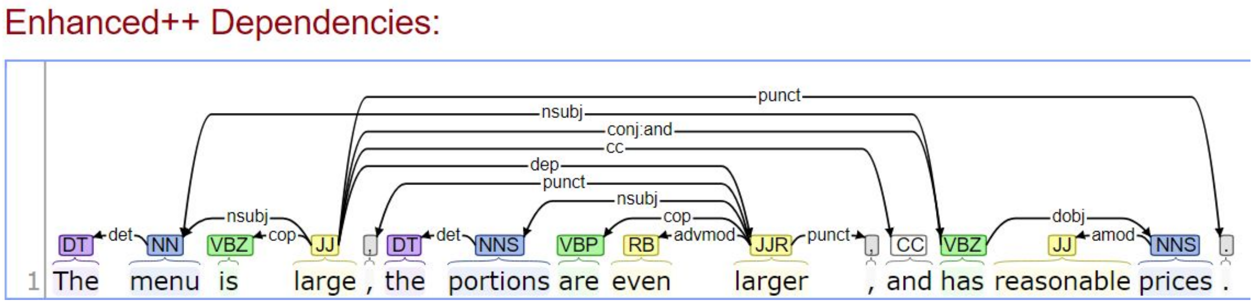
This assignment focuses on natural language understanding. Assume a user, Lily, wants to decide whether or not to have an important dinner at Union Grill on Craig Street. She wants to know the past consumers' opinion on the restaurant. For example, does Union Grill provide good service? Does it have delicious food? Does it provide a good price for the food? You want to help her by mining Yelp, which contains 368 reviews about the restaurant. **Opinion** is an aspect of the restaurant in Yelp's reviewer assessment.

You decide to design and develop a reviewer opinion extraction and querying system for this task: (1) the system automatically extracts opinions from all the reviews; (2) the user inputs an opinion as a query, and your system compares the input opinion with the extracted opinions and returns similar opinions (3) a set of reviews are returned as supporting evidence for this opinion.

Dataset: You are provided with 20 reviews randomly selected from Yelp of Union Grill. We do know you will face a large amount of data in a real scenario, but manually annotation is expensive, so we only provide 20 reviews in this assignment. You can regard it as a validation dataset, i.e., design and tune your system to get best performance on this small corpus. The IDs for these 20 reviews are from 1 to 20.

Task 1: Extract opinions from reviews with CoreNLP tool.

The extracted opinions from the reviews can be represented as a tuple with two elements: attribute and value. The attribute can be an aspect of the restaurant, i.e., an entity or a few entities of the restaurant. The value is a descriptive word that provides the assessment mentioned in the reviews. For example, three opinions are expressed in reviews **"The menu is large, the portions are even larger, and the prices are reasonable.",** the system should extract three tuples **[menu, large], [portion, large], [price, reasonable],** and menu, portion and price are attributes, large, large and reasonable are values. Stanford coreNLP (demo URL) will provide parsing in the following figure, and maybe you want to extract nsubj(large, menu), nsubj(larger, portions) and amod(prices, reasonable). The examples we provide here are only some forms of opinions.You are strongly encouraged to read the reviews to design your own ways to extract opinions based on the coreNLP Enhanced Dependencies Annotation parsing results.


Enhanced++ Dependencies:
punct
nsubj
conj:and
cc
dep
punct
nsubj
cop
advmod
nsubj
cop
det
det
punct
dobj
amod
DT NN VBZ JJ . DT NNS VBP RB JJR . CC VBZ JJ NNS .
1 The menu is large , the portions are even larger , and has reasonable prices .

Download CoreNLP tool: Python.
Materials (but not limited to) to help you understand and use CoreNLP:
https://interviewbubble.com/stanford-corenlp-tutorial/
https://nlp.stanford.edu/software/dependencies_manual.pdf
https://en.wikipedia.org/wiki/Brown_Corpus#Part-of-speech_tags_used
https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.9.8216&rep=rep1&type=pdf

Task 2: Given the input opinion as the query, find the similar extracted opinions.

User will input an opinion as the query, and your system compares it against the extracted opinions. For example, in your extracted opinions, there are “**excellent service**”, “**nice waiter**”, “**great service**”, and “**good service**”. When Lily inputs “**good service**”, your system should return all these extracted opinions since they are all the evidence supporting “good service”.

The semantic similarity between different words can be obtained by word embedding. You can use Google pre-trained word embeddings [URL] with Skip-gram model, and use cosine similarity to measure the word semantic similarity. You need to tune the threshold for cosine similarity (***cosine_sim*** *in Assignment4Main*) to determine whether words are similar or not. However, the Google pre-trained word embeddings is too large, 8G after decompression. Therefore, we prepared assign4_word2vec1.bin, which only contains words that appear in our corpus. We provided example word2vec codes in ***FindSimilarOpinions*** for your reference.

After finding the similar opinions, you need to return the review IDs. Materials (but not limited to) to help you process and understand the word embedding vectors are:
https://deeplearning4j.org/docs/latest/deeplearning4j-nlp-word2vec
https://arxiv.org/pdf/1301.3781.pdf%5D

Please note that there is no correct solution to this assignment. You need to understand the task, learn the NLP tools, and try to make your algorithm performance better by tuning the threshold.

Suggested ground-truth for 4 query opinions for your reference:
According to **manually annotation**, we provide ground_truth review_IDs for the four input query opinion:

query opinion [**service, good**] has similar opinions:
[service, excellent] appears in review 1, 2
[service, great] appears in review 5, 14
[service, warm] appears in review 8
[service, solid] appears in review 13
[service, good] appears in review 20
[waiter, kind] appears in review 16
[waiter, friendly] appears in review 17
[waiter, attentive] appears in review 17

query opinion [**service, bad**] has similar opinions:
[server, rude] appears in review 4
[service, rude] appears in review 7
[service, bad] appears in review 9
[service, slow] appears in review 19
[waiter, slow] appears in review 15

query opinion [**atmosphere, good**] has similar opinions:
[atmosphere, nice] appears in review 3, 4
[atmosphere, great] appears in review 5
[atmosphere, fun] appears in review 14
[feeling, warm] appears in review 12
[ambience, pleasant] appears in review 20

query opinion [**food, delicious**] has similar opinions:
[meal, delicious] appears in review 4
[food, great] appears in review 5
[food, excellent] appears in review 6
[food, hearty] appears in review 8
[food, excellent] appears in review 13
[food, fresh] appears in review 14
[food, interesting] appears in review 14
[food, satisfying] appears in review 16
[food, fresh] appears in review 18
If you are interested in mining deeper in this area, you may also find:
[bread stick, delicious] appears in review 1
[california salad, delicious] appears in review 1
[meat, flavorful] appears in review 2

[meat, tender] appears in review 2, 18
[potatoes, yum] appears in review 2
[cole slaw, delicious] appears in review 2
[waffle fries, amazing] appears in review 3
[turkey devonshire, awesome] appears in review 6
[fish taco, good] appears in review 15
[nachos, good] appears in review 17
[food quality, excellent] appears in review 11
[waffles fries, great] appears in review 17

Coding:

Please install the package "gensim" for vector similarity calculation.

Please feel free to add extra python files, functions, attributes to do training, validation, and testing. Your grading will be based on the output of running Assignment4Main and the report. **Assignment4Main** is the main class for running your assignment, which **cannot** be modified.

Report:

You need to clearly write:

(1) How did you design your opinion extraction module with CoreNLP?
(2) How did you measure the opinion similarity? How do you tune the threshold?
(3) Discuss the successful cases that your system can handle.
(4) Discuss the cases that your system fail. For example, you may find the review shows an opinion similar to the input opinion, but your algorithm fail to extract. Or you may have a similar opinion, but cannot be matched to the input opinion through your current algorithm.
(5) The output of your algorithm with your best results (threshold).

Grading:

Your submission will be graded based on:

1. Correctness of the implementation on two tasks (40%)
2. Performance of your algorithms. A good algorithms should provide expected review_ids, as listed in **Assignment4Main**, as many as possible. (20%)
3. A clear report in **PDF** format (40%)

Submission Requirements

A zipped file package with the naming convention as "pittids_a4". For example, suppose the Pitt id is jud1, then the submission package should be jud1_a4.zip.

The file package should contain:

1. All the scripts/programs you used for this assignment. (**src folder**)
2. A clear report. (**pdf file**)
3. Your output of Assignment4Main. (This should also be included in the **pdf file**.)
4. The tool you used in the assignment. (This should also be included in the **pdf file**.)

Do not upload the stanford CoreNLP package file and the pre-trained word2vec.