Name: Kang Li
Student Number: 14203445

Assignment 2 Guide

Opinion Mining

The main method is in the class ***Execute.java* from the package ***alg*** **

The procedures of this assignment can be described as the following:

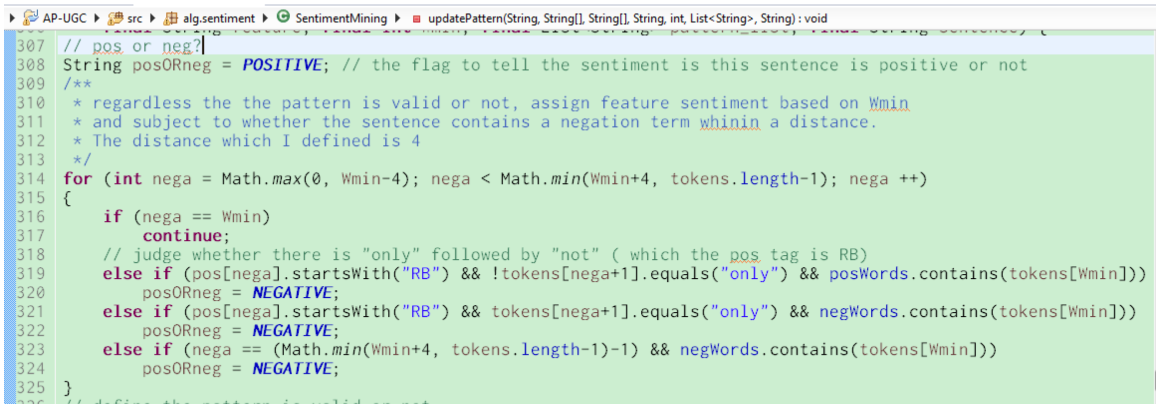
- Initialize the dataset: Read in the original dataset (sentiment lexicon, product features, the same meaning features…)
- Sentiment mining: Extract the features (bi-gram feature or single-noun feature) and identify the sentiment word (Wmin), then convert any words that occur between this feature and Wmin into pattern.
- Opining mining: Judge the nationality of the patterns and assign feature sentiment, then create the experience case representation.
- Summary: Create a summary table for each product
- Comparison: compare the given product based on the common features' overall sentiment value.
- Recommendation: Recommend products which are similar but better to the users' queries based on the combination between cosine similarity and sentiment.

Task 1

Firstly, a new class ***FileReader.java*** was created in the package ***util.reader***, which will read the sentiment lexicon and feature name from the original given dataset, as well as the same meaning features from the folder ***same features*** created by me.

Then, extracting features and sentiment mining will be processed in the new class ***SentimentMining.java*** in the package ***alg.sentiment***. Besides, regardless the pattern is valid or not, defining the feature sentiment will be handled in this class, which avoids to read the ***DatasetReader*** again. The surprise result is that the whole process from the dataset reader until recommendation generation only cost within 90 second for the Printer Category, even though the feature sentiment did not store to the DATABASE.

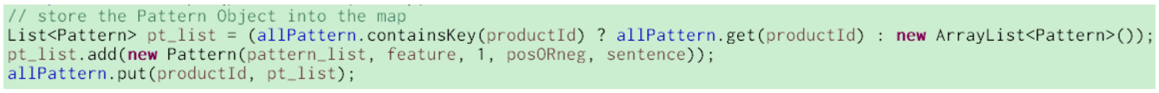
The following screenshot is the method about defining the feature sentiment within a 4-word-distance either side of Wmin:



AP-UGC ▸ src ▸ alg.sentiment ▸ SentimentMining ▸ updatePattern(String, String[], String[], String, int, List<String>, String) : void

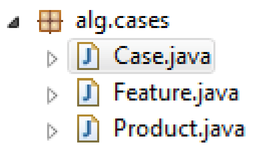
```
// pos or neg?
String posORneg = POSITIVE; // the flag to tell the sentiment is this sentence is positive or not
/**
 * regardless the the pattern is valid or not, assign feature sentiment based on Wmin
 * and subject to whether the sentence contains a negation term whinin a distance.
 * The distance which I defined is 4
 */
for (int nega = Math.max(0, Wmin-4); nega < Math.min(Wmin+4, tokens.length-1); nega ++)
{
    if (nega == Wmin)
        continue;
    // judge whether there is "only" followed by "not" ( which the pos tag is RB)
    else if (pos[nega].startsWith("RB") && !tokens[nega+1].equals("only") && posWords.contains(tokens[Wmin]))
        posORneg = NEGATIVE;
    else if (pos[nega].startsWith("RB") && tokens[nega+1].equals("only") && negWords.contains(tokens[Wmin]))
        posORneg = NEGATIVE;
    else if (nega == (Math.min(Wmin+4, tokens.length-1)-1) && negWords.contains(tokens[Wmin]))
        posORneg = NEGATIVE;
}
```

All the information will be stored into the Pattern Object in a Map container:

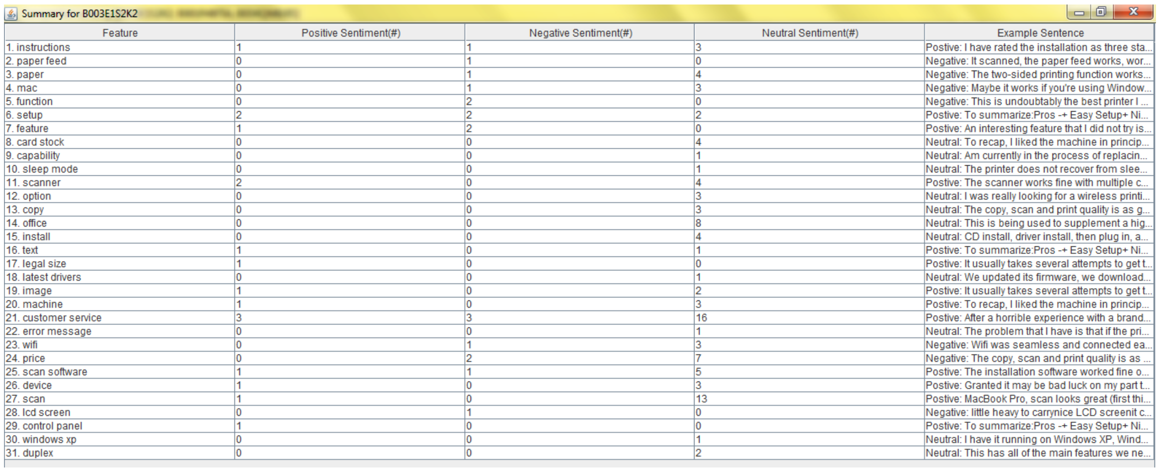


```
// store the Pattern Object into the map
List<Pattern> pt_list = (allPattern.containsKey(productId) ? allPattern.get(productId) : new ArrayList<Pattern>());
pt_list.add(new Pattern(pattern_list, feature, 1, posORneg, sentence));
allPattern.put(productId, pt_list);
```

Moreover, there is a Feature Object in the Pattern Object, and those two Object class can be found in the package ***alg.cases***. Also, both of them implements the ***Case Interface*** created in the same package:



Finally, all of the pattern object converted from ***SentimentMining.java*** will be passed to the ***Summary.java*** in the package ***alg.summary***. In order to make the summary table clear, the class **JTable** is used to display the table:

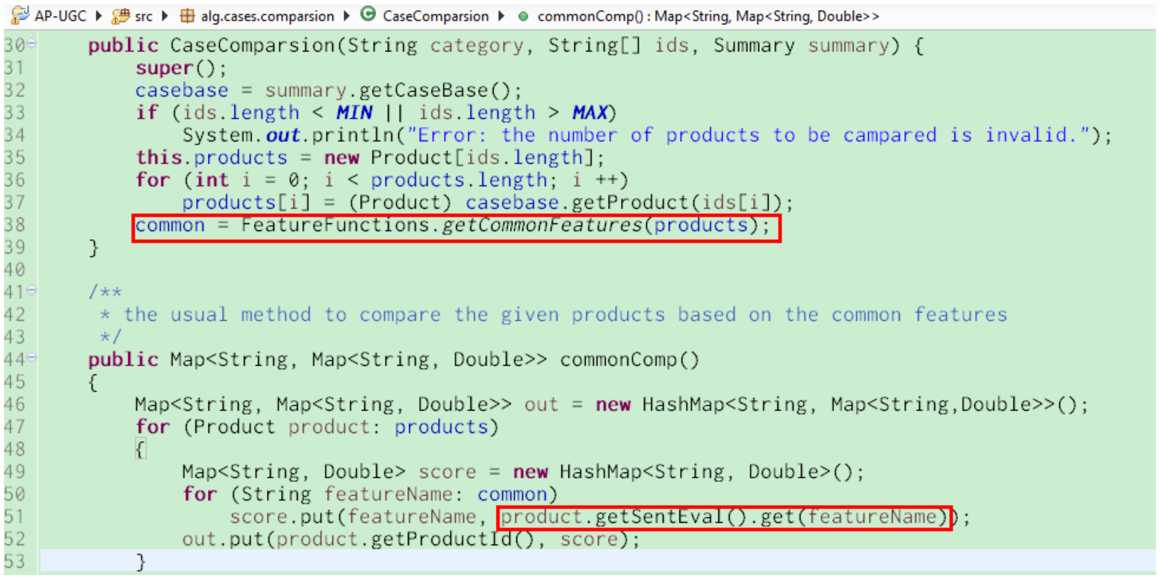


Summary for B003E1S2K2

Feature	Positive Sentiment(#)	Negative Sentiment(#)	Neutral Sentiment(#)	Example Sentence
1. instructions	1	1	3	Postive: I have rated the installation as three sta...
2. paper feed	0	1	0	Negative: It scanned, the paper feed works, wor...
3. paper	0	1	4	Negative: The two-sided printing function works...
4. mac	0	1	3	Negative: Maybe it works if you're using Window...
5. function	0	2	0	Negative: This is undoubtably the best printer I ...
6. setup	2	2	2	Postive: To summarize:Pros -+ Easy Setup+ Ni...
7. feature	1	2	0	Postive: An interesting feature that I did not try is...
8. card stock	0	0	4	Neutral: To recap, I liked the machine in princip...
9. capability	0	0	1	Neutral: Am currently in the process of replacin...
10. sleep mode	0	0	1	Neutral: The printer does not recover from slee...
11. scanner	2	0	4	Postive: The scanner works fine with multiple c...
12. option	0	0	3	Neutral: I was really looking for a wireless printi...
13. copy	0	0	3	Neutral: The copy, scan and print quality is as g...
14. office	0	0	8	Neutral: This is being used to supplement a hig...
15. install	0	0	4	Neutral: CD install, driver install, then plug in, a...
16. text	1	0	1	Postive: To summarize:Pros -+ Easy Setup+ Ni...
17. legal size	1	0	0	Postive: It usually takes several attempts to get t...
18. latest drivers	0	0	1	Neutral: We updated its firmware, we download...
19. image	1	0	2	Postive: It usually takes several attempts to get t...
20. machine	1	0	3	Postive: To recap, I liked the machine in princip...
21. customer service	3	3	16	Postive: After a horrible experience with a brand...
22. error message	0	0	1	Neutral: The problem that I have is that if the pri...
23. wifi	0	1	3	Negative: Wifi was seamless and connected ea...
24. price	0	2	7	Negative: The copy, scan and print quality is as ...
25. scan software	1	1	5	Postive: The installation software worked fine o...
26. device	1	0	3	Postive: Granted it may be bad luck on my part t...
27. scan	1	0	13	Postive: MacBook Pro, scan looks great (first thi...
28. lcd screen	0	1	0	Negative: little heavy to carrynice LCD screenit c...
29. control panel	1	0	0	Postive: To summarize:Pros -+ Easy Setup+ Ni...
30. windows xp	0	0	1	Neutral: I have it running on Windows XP, Wind...
31. duplex	0	0	2	Neutral: This has all of the main features we ne...

Task 2

In task2, I created a new package ***alg.cases.comparsion***, which contains the class ***CaseComparsion.java*** and ***Comparsion.java***.



AP-UGC ▸ src ▸ alg.cases.comparsion ▸ CaseComparsion ▸ commonComp() : Map<String, Map<String, Double>>

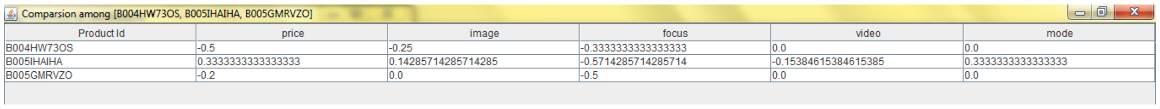
```
    public CaseComparsion(String category, String[] ids, Summary summary) {
        super();
        casebase = summary.getCaseBase();
        if (ids.length < MIN || ids.length > MAX)
            System.out.println("Error: the number of products to be campared is invalid.");
        this.products = new Product[ids.length];
        for (int i = 0; i < products.length; i ++)
            products[i] = (Product) casebase.getProduct(ids[i]);
        common = FeatureFunctions.getCommonFeatures(products);
    }

    /**
     * the usual method to compare the given products based on the common features
     */
    public Map<String, Map<String, Double>> commonComp()
    {
        Map<String, Map<String, Double>> out = new HashMap<String, Map<String,Double>>();
        for (Product product: products)
        {
            Map<String, Double> score = new HashMap<String, Double>();
            for (String featureName: common)
                score.put(featureName, product.getSentEval().get(featureName));
            out.put(product.getProductId(), score);
        }
```

Considering that the feature attribute will be handled frequently (for example, to get the common features or union features among a product list), the Static Methods were created in the ***class FeatureFunction.java*** from the package ***util***.

To be precise, the comparison is based on the common features' overall sentiment value, and the equation is defined as following:

$$Sent(F_i, P) = \frac{Pos(F_i, P) - Neg(F_i, P)}{Pos(F_i, P) + Neg(F_i, P) + Neut(F_i, P)}$$

Again, the comparison table can be obtained used the class JTable.java:

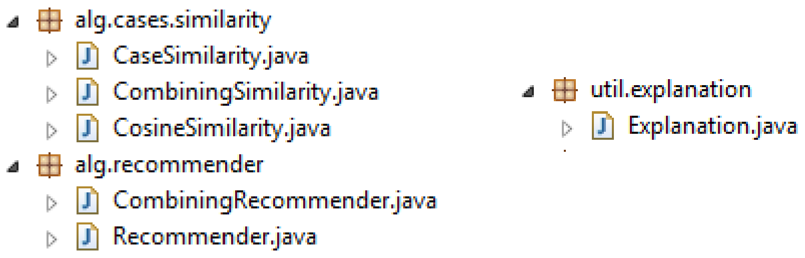


Comparsion among [B004HW73OS, B005IHAIHA, B005GMRVZO]

Product Id	price	image	focus	video	mode
B004HW73OS	-0.5	-0.25	-0.3333333333333333	0.0	0.0
B005IHAIHA	0.3333333333333333	0.14285714285714285	-0.5714285714285714	-0.15384615384615385	0.3333333333333333
B005GMRVZO	-0.2	0.0	-0.5	0.0	0.0

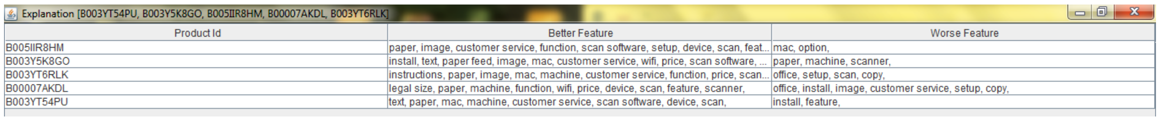
Task 3

This task aims to recommend a top-N list of product that are similar and better compared to the target product, and then explain the reasons for the top-N list product. Hence, we should calculate the similarity based on Cosine and Sentiment, so the package ***alg.cases.similarity*** was created; in order to generate the recommendation list, I created a new package ***alg.recommender***; the next step is to explain the reasons, so the new package ***util.explanation*** was created:



```
alg.cases.similarity
    CaseSimilarity.java
    CombiningSimilarity.java
    CosineSimilarity.java
alg.recommender
    CombiningRecommender.java
    Recommender.java

util.explanation
    Explanation.java
```

Finally, the explanation table can be created by the **JTable.java**:

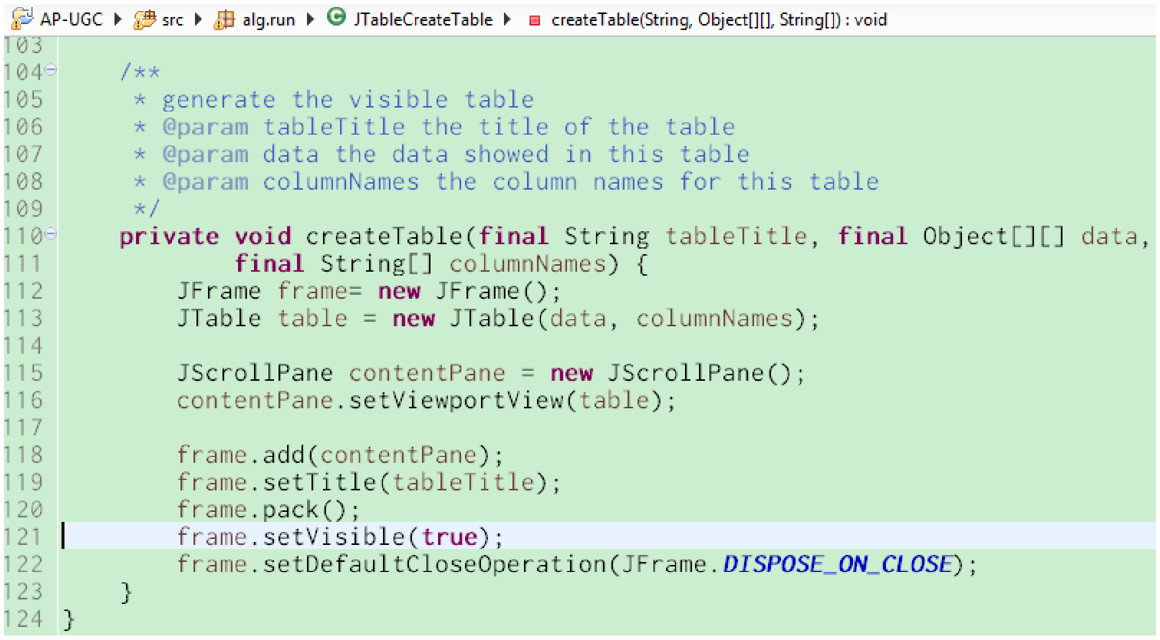


Explanation [B003YT54PU, B003Y5K8GO, B005IIR8HM, B00007AKDL, B003YT6RLK]

Product Id	Better Feature	Worse Feature
B005IIR8HM	paper, image, customer service, function, scan software, setup, device, scan, feat...	mac, option,
B003Y5K8GO	install, text, paper feed, image, mac, customer service, wifi, price, scan software, ...	paper, machine, scanner,
B003YT6RLK	instructions, paper, image, mac, machine, customer service, function, price, scan...	office, setup, scan, copy,
B00007AKDL	legal size, paper, machine, function, wifi, price, device, scan, feature, scanner,	office, install, image, customer service, setup, copy,
B003YT54PU	text, paper, mac, machine, customer service, scan software, device, scan,	install, feature,

----

A new class ***JTableCreateTable.java*** was created in the package ***alg***, which will generate a new window to show the table. In addition, the creating table method's screenshot is attached:



AP-UGC ▸ src ▸ alg.run ▸ JTableCreateTable ▸ createTable(String, Object[][], String[]) : void

```java

    /**
     * generate the visible table
     * @param tableTitle the title of the table
     * @param data the data showed in this table
     * @param columnNames the column names for this table
     */
    private void createTable(final String tableTitle, final Object[][] data,
            final String[] columnNames) {
        JFrame frame= new JFrame();
        JTable table = new JTable(data, columnNames);

        JScrollPane contentPane = new JScrollPane();
        contentPane.setViewportView(table);

        frame.add(contentPane);
        frame.setTitle(tableTitle);
        frame.pack();
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    }
}
```