Jigar Rudani (rudani2)
CS 412 Data Mining

Assignment 3 Frequent Pattern Mining Programming (Fall 2014)

Algorithm

Step 1: Mining Frequent Patterns for Each Topic

Apriori pattern mining

1. Initially we consider each item in the transaction and generate Candidate C1 item set.
2. There are 2 important steps in Apriori pattern mining
    a. Prune Step
    b. Join Step
3. After generating C1 item set. We apply Prune step to it and generate L1 itemset
4. We perform Join step for each Lk itemset.
5. After Join step we generate remaining Ck item set (K >= 2) by applying Prune step to each item generated.
6. Repeat step 4 to step 5 till Ck = Null

Prune Step

1. We calculate the frequency of each item by counting the number of occurrences in the topic file.
2. We then compare frequency with min_support.
3. If frequency > min_support then
    a. Include the item in Frequent K Item Set
    Else
    a. Excludes item
4. We can check the Frequency of K (K > 1) itemset by checking whether each subset of itemset present in Lk-1 Itemset.
5. I have used min_support as 0.01 which I have explained in below step

Join Step

1. We perform self join operation on each Lk itemset.
    a. E.g [1,2] Join [1,3] → [1,2,3]

Output Path

1. I have created generalize method which helps to print desired output to each respective file in each respective folder.

Source Code →

1. Python File → ***rudani2_apriori.py***
2. Method Name → a. ***funcAprioriFreqPattern()***
    b. func_PruneItems()
    c. func_JoinItems()
    d. func_redirecttoOutputFolder()

Step 2: Mining Maximal/Closed Patterns

Closed Pattern Mining

1. Closed Patterns are patterns which doesn't have proper superset. If it has proper superset then frequency should not be same.

Source Code →
1. Python File → ***rudani2_apriori.py***
2. Method Name → a. ***func_ClosedMaxPattern()***

Maximal Pattern Mining

1. Maximal Patterns are pattern which doesn't have superset.
2. All Maximal patterns are expected to be closed as well
3. In order to optimize the code I have created same method for Closed and Max as finding superset operation is common to both

Source Code →
1. Python File → ***rudani2_apriori.py***
2. Method Name → a. ***func_ClosedMaxPattern()***

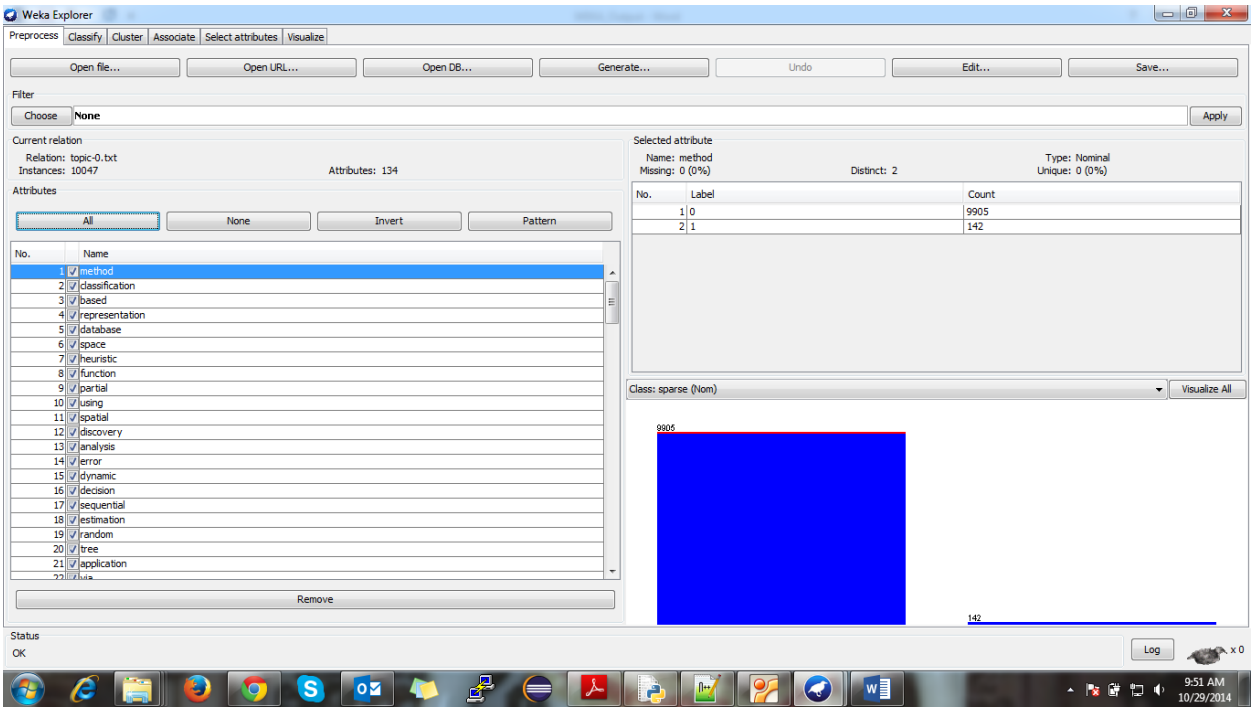
Step 3: Association Rule Mining by Weka

1. Genrated .arff files for each topics. Below are the screen shots for it

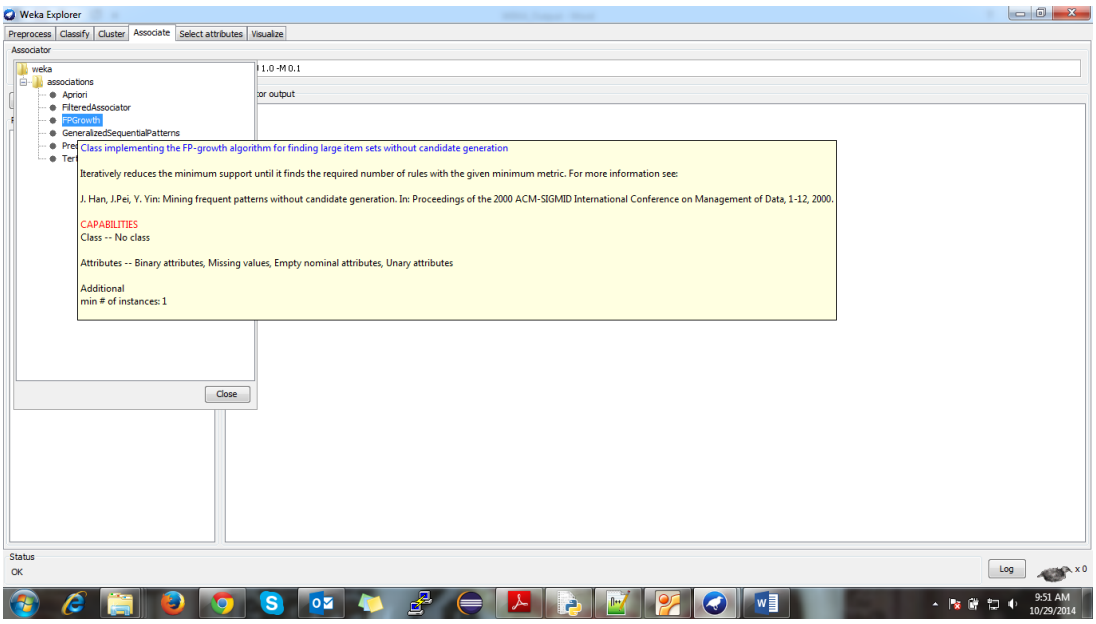
WEKA

Filename: - topic-0.arff

Preprocess →



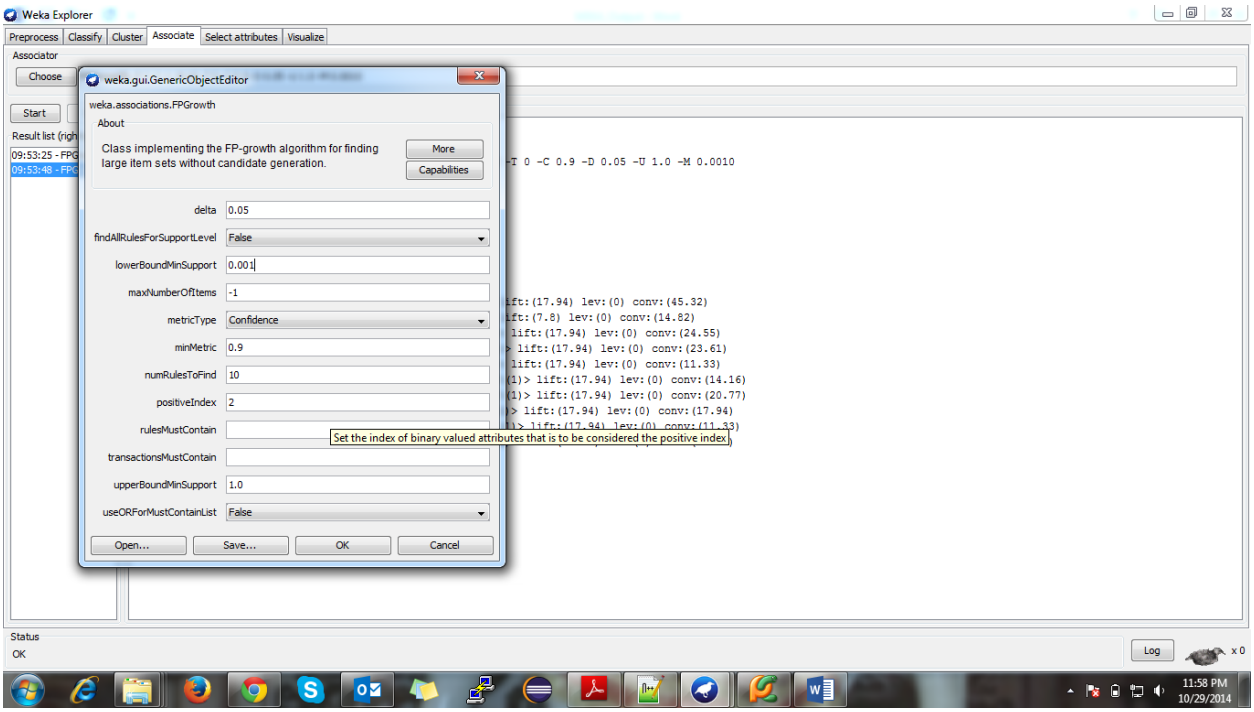
Associator Choose → FP GROWTH



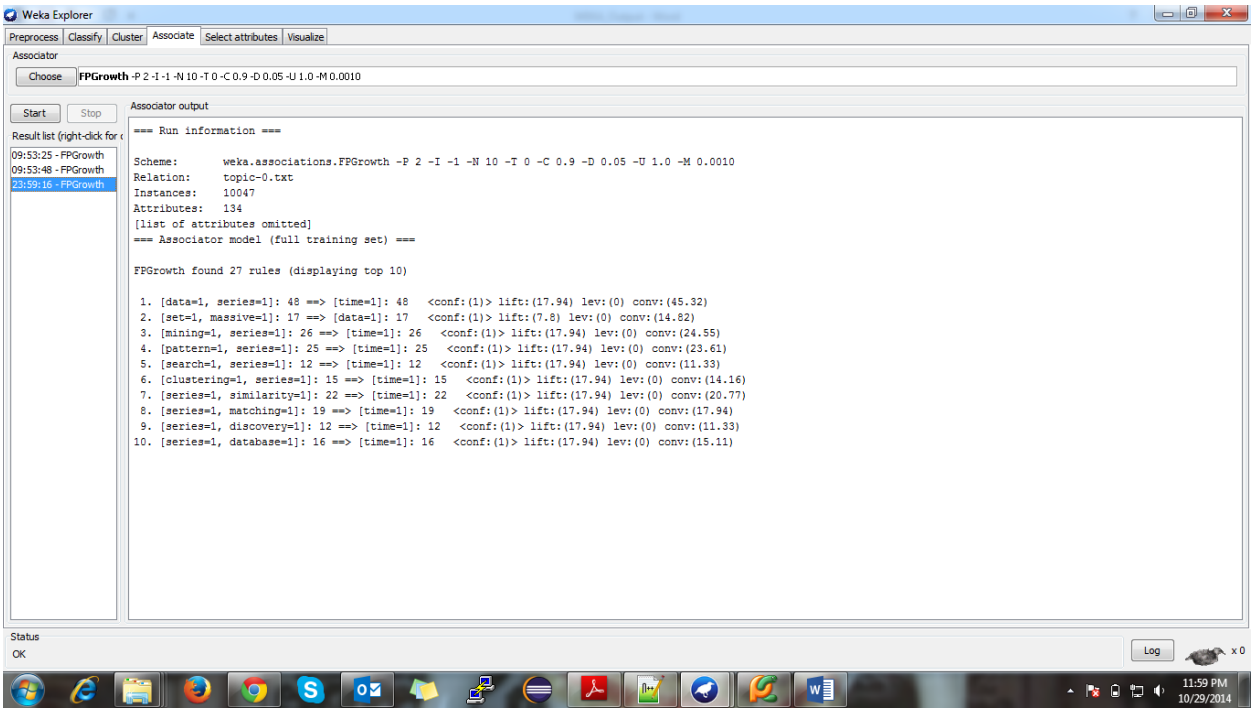
Parameter Settings →

lowerBoundMinSupport → 0.001

minMetric → 0.9

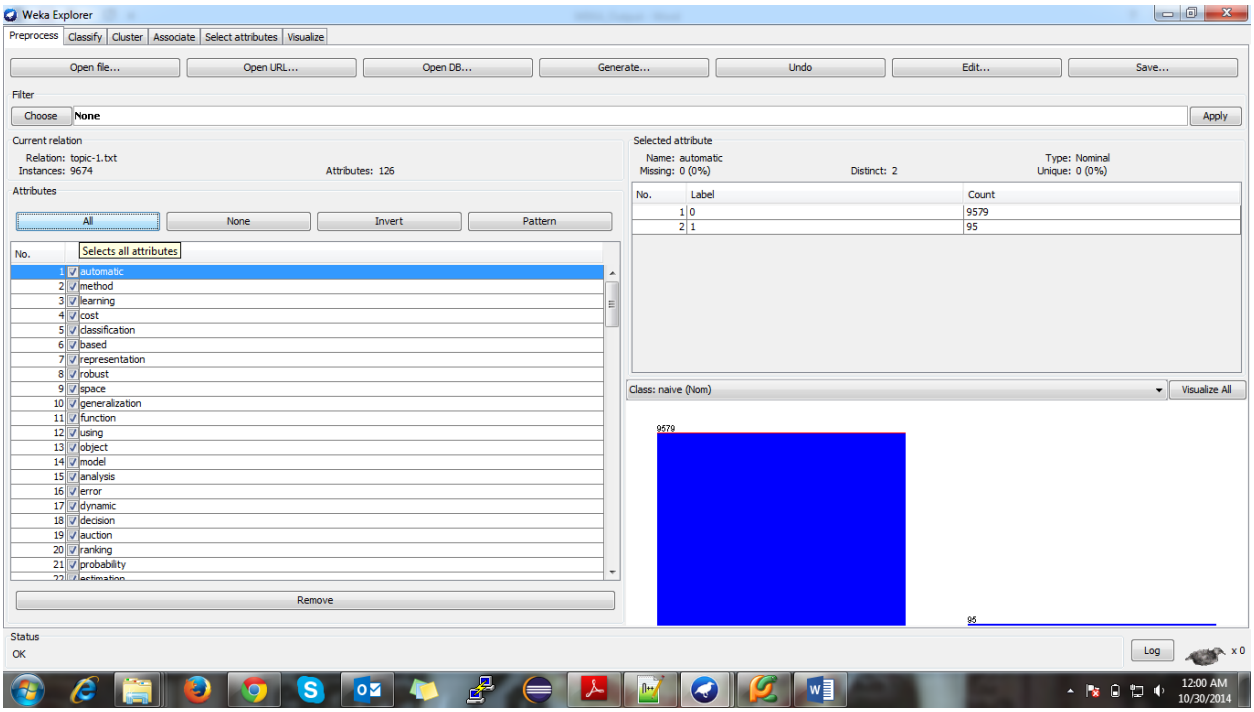


Association Rules →



Filename: - topic-1.arff

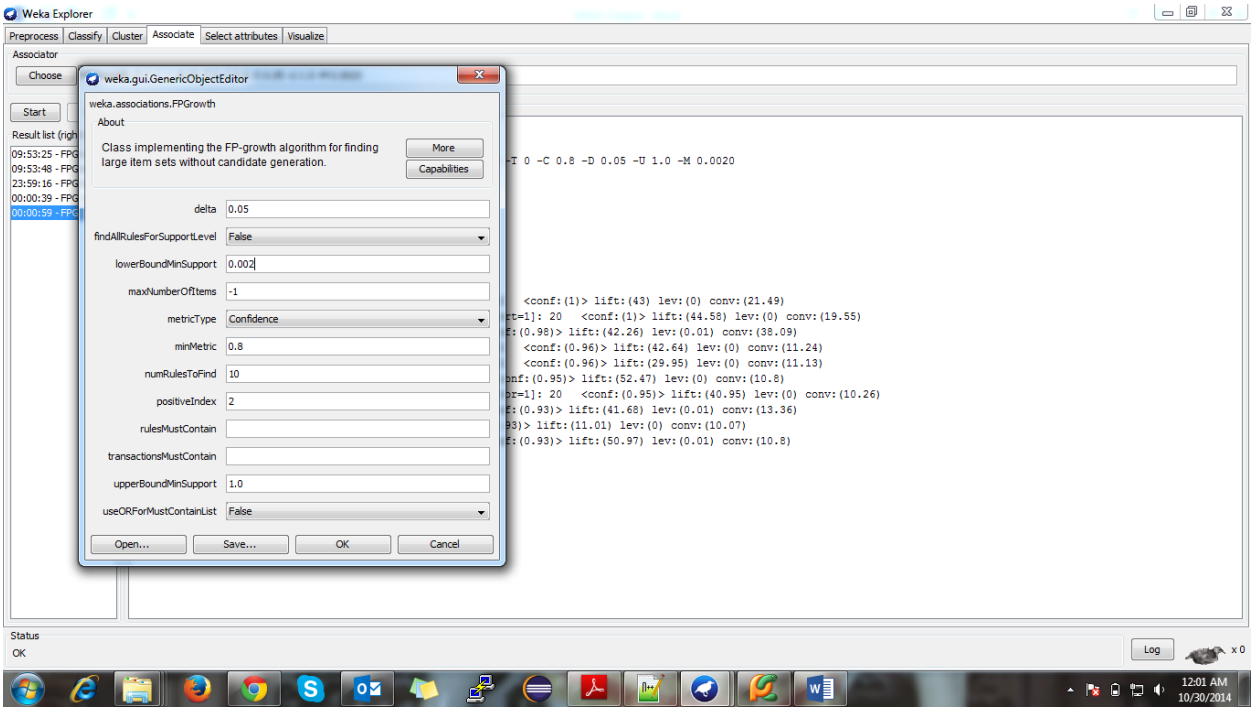
Preprocess →



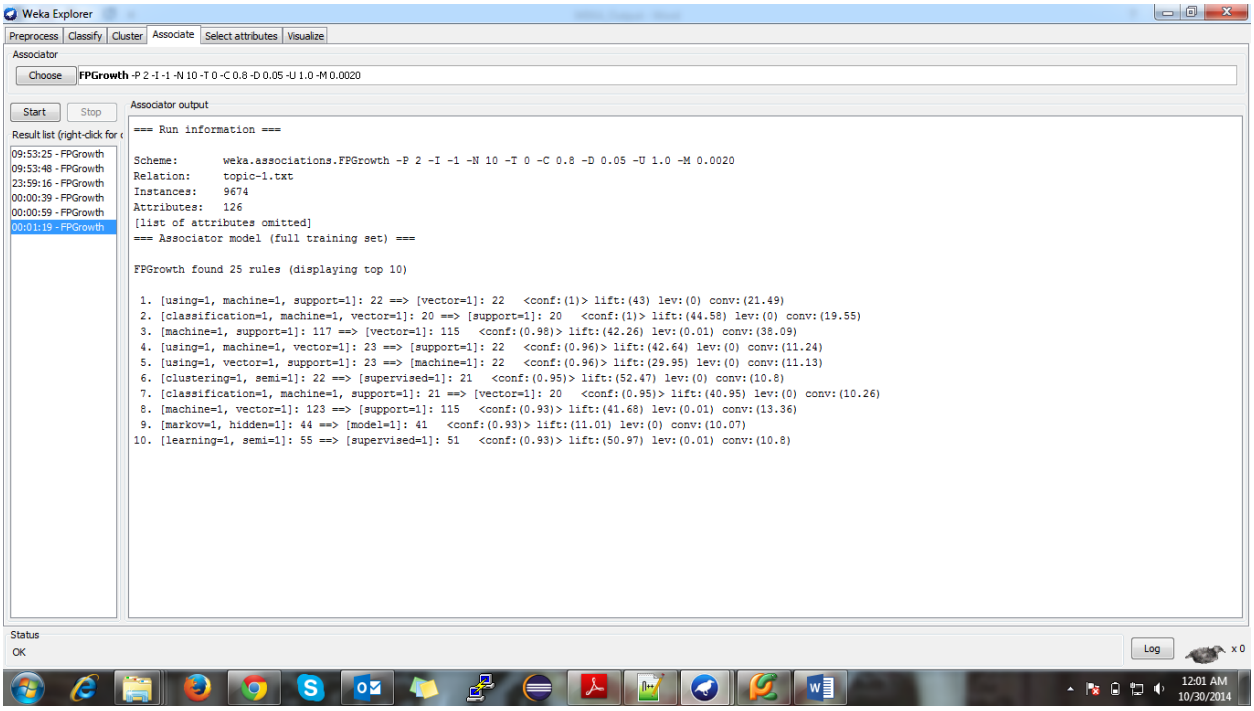
Parameter Settings →

lowerBoundMinSupport → 0.002

minMetric → 0.8

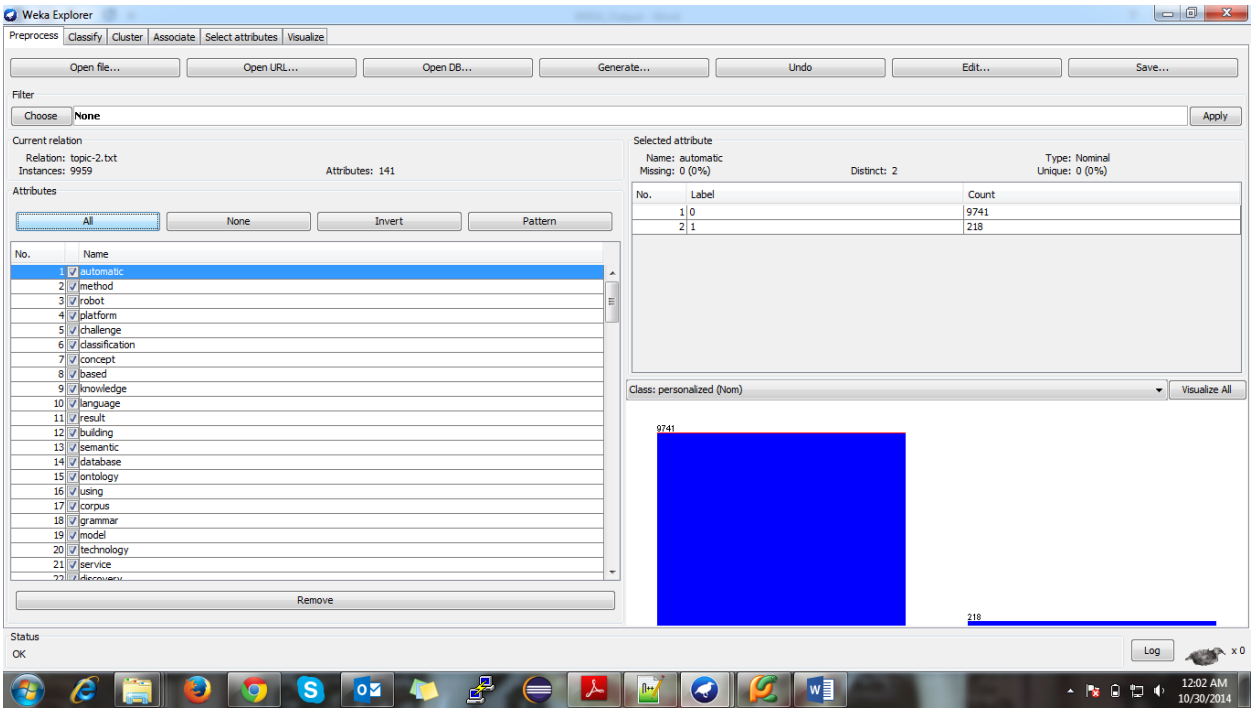


Association Rules →



Filename: - topic-2.arff

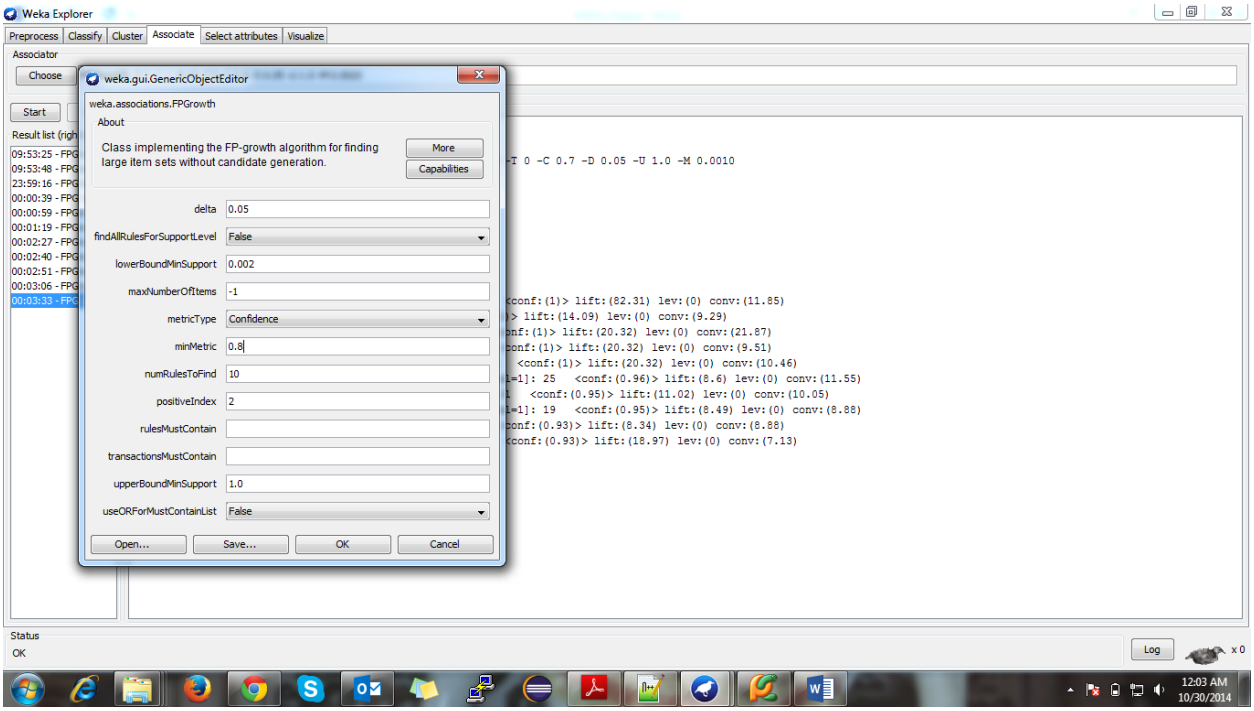
Preprocess →



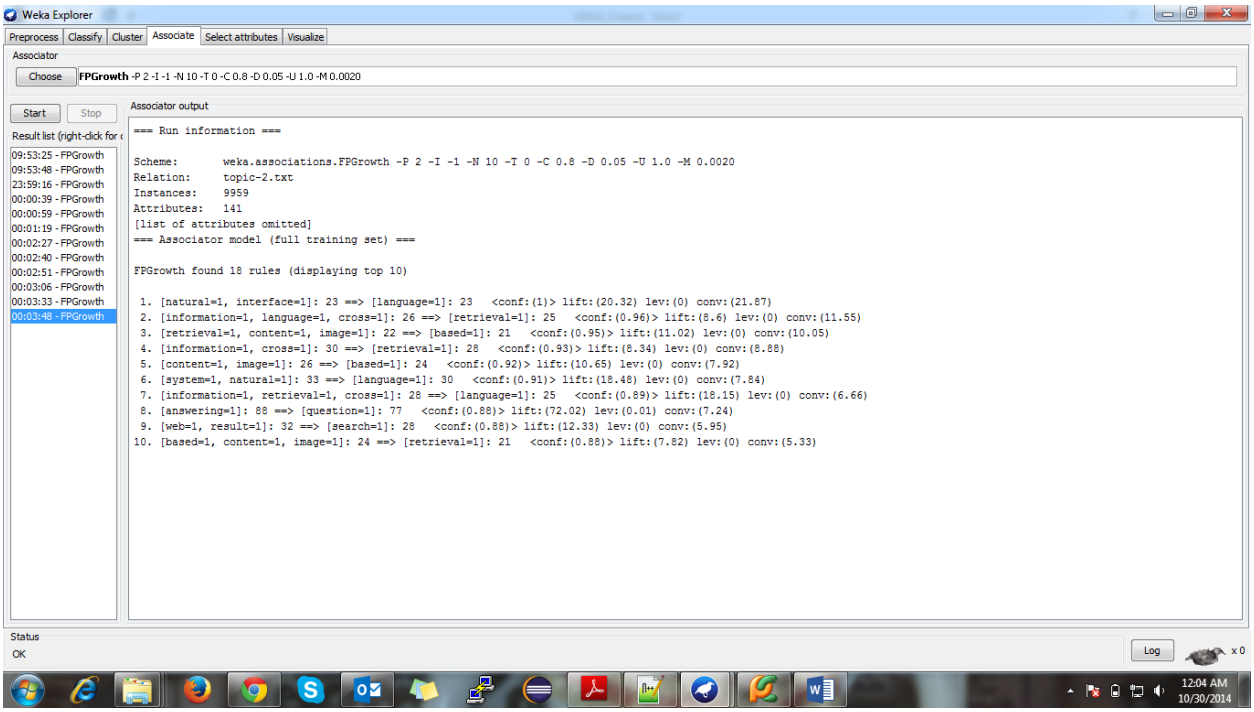
Parameter Settings →

lowerBoundMinSupport → 0.002

minMetric → 0.8

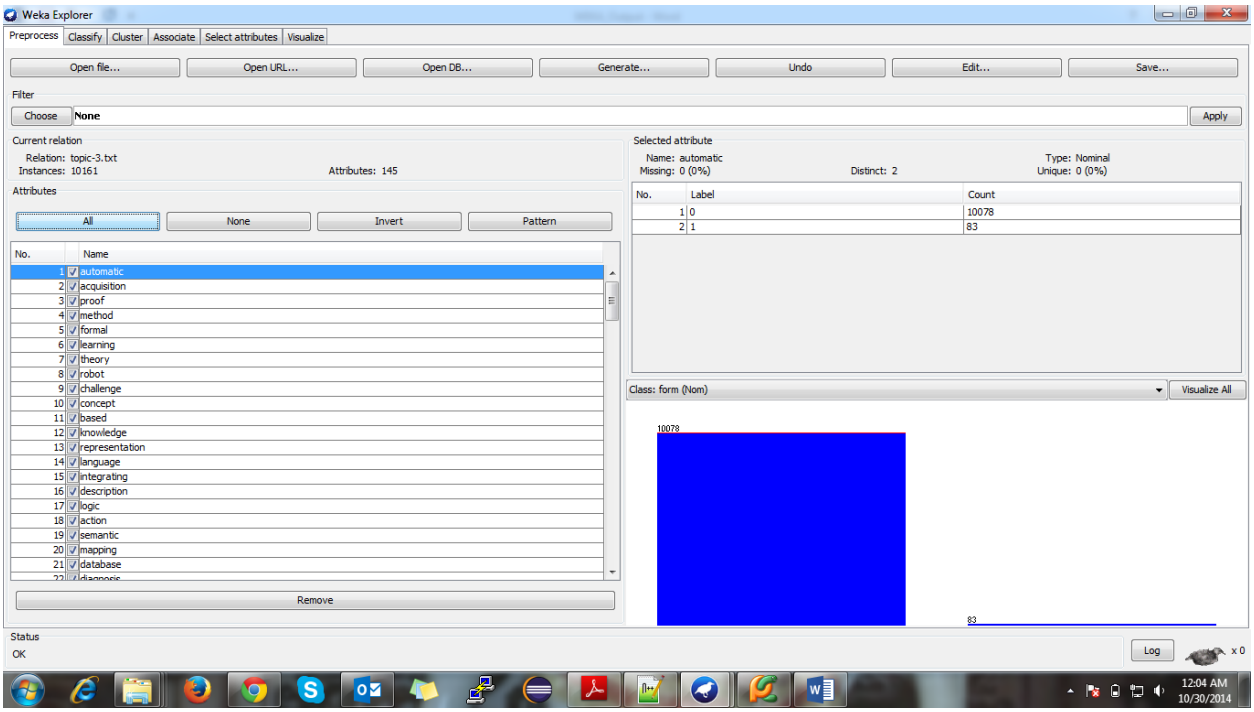


Association Rules →



Filename: - topic-3.arff

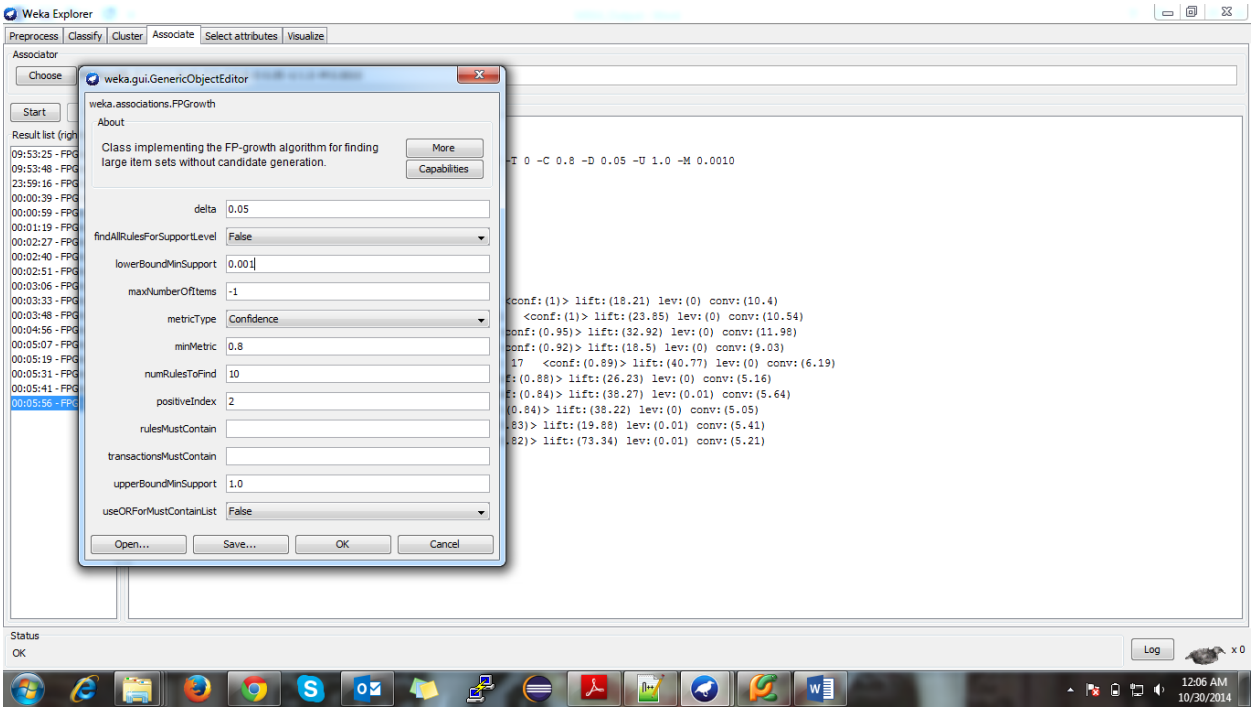
Preprocess →



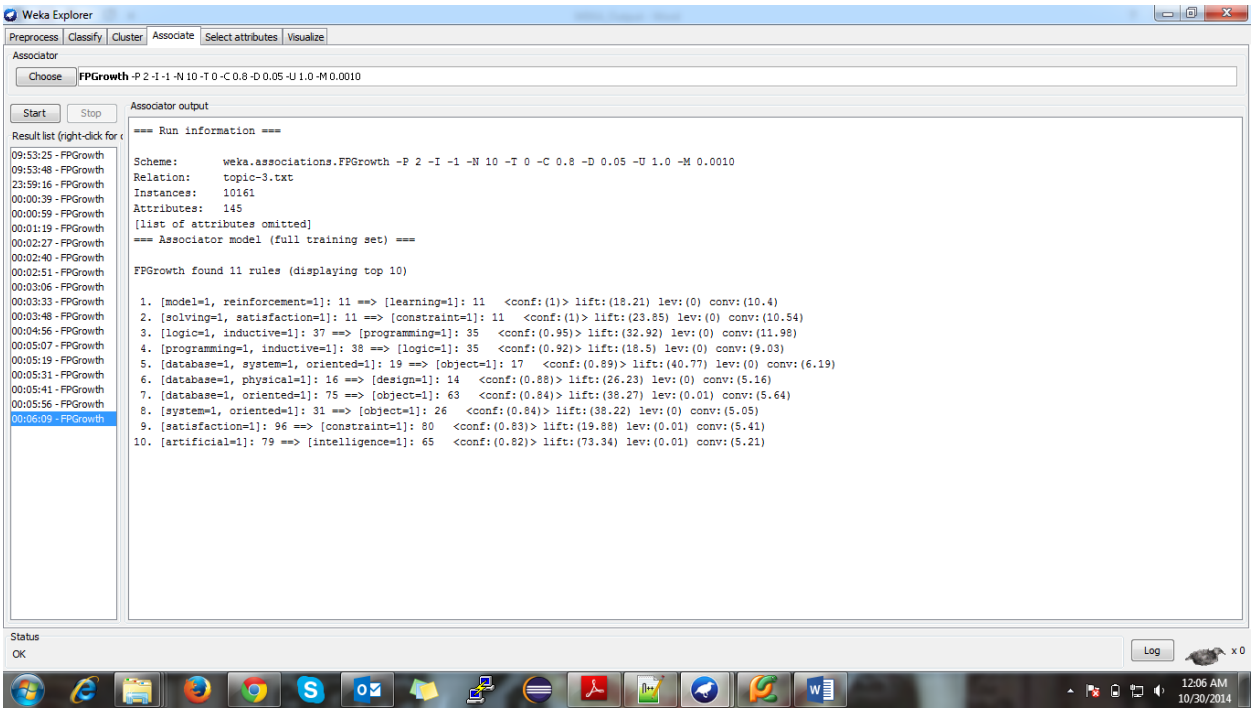
Parameter Settings →

lowerBoundMinSupport → 0.001

minMetric → 0.8

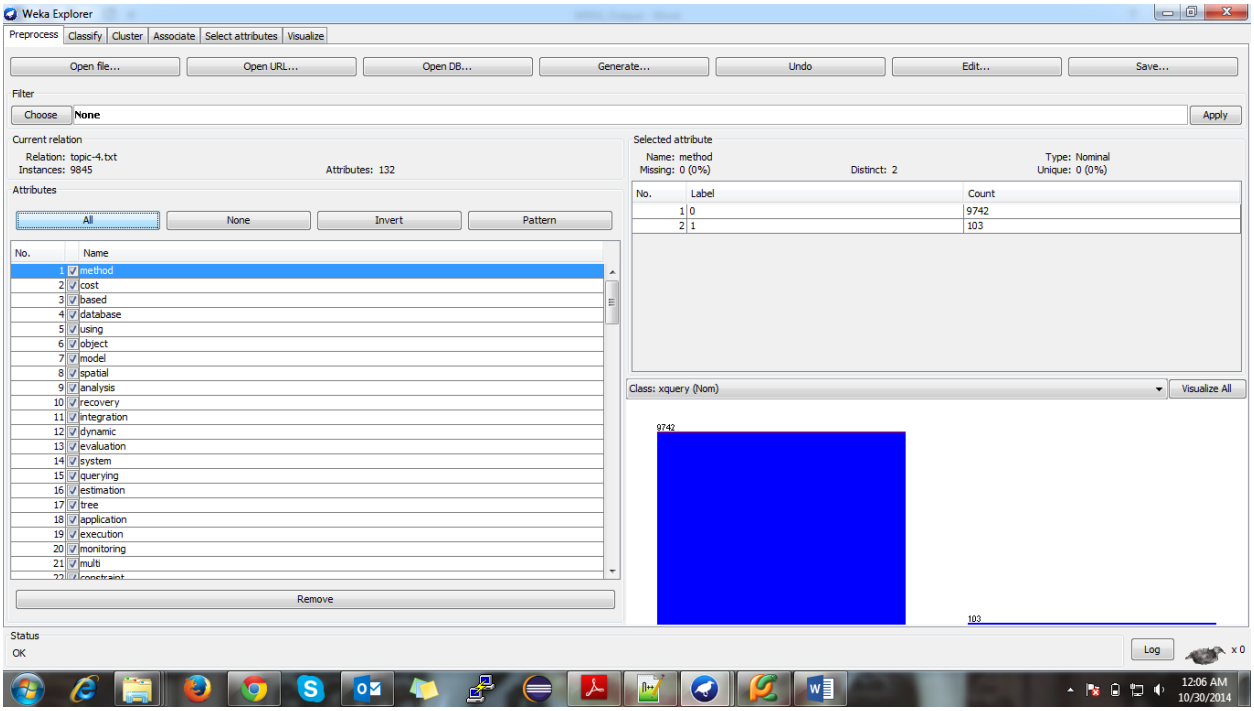


Association Rules →



Filename: - topic-4.arff

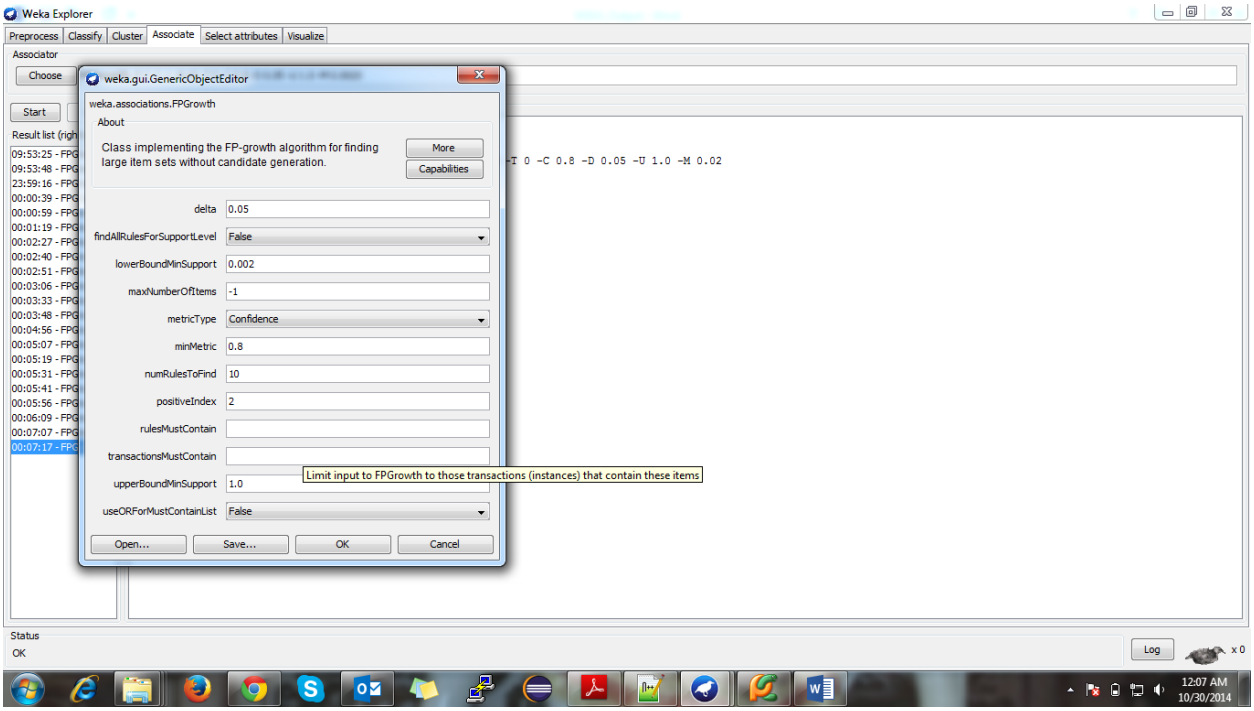
Preprocess →



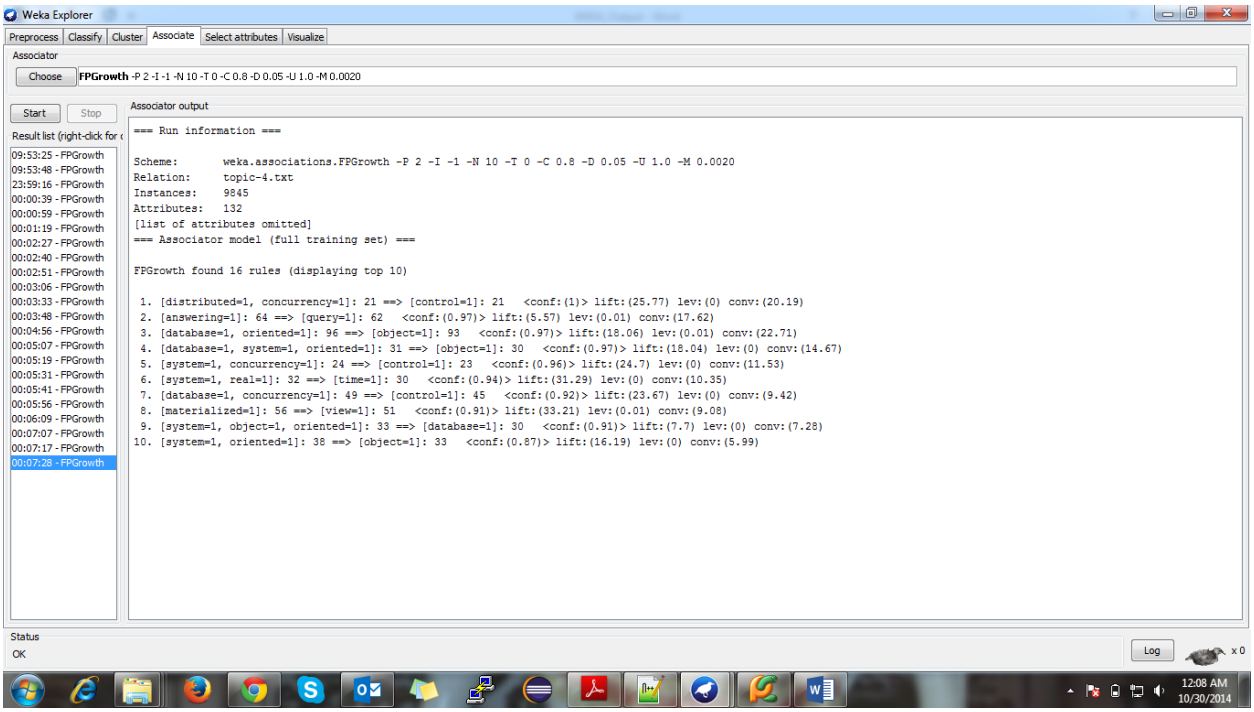
Parameter Settings →

lowerBoundMinSupport → 0.002

minMetric → 0.8



Association Rules →



Step 4: Re-rank by Purity of Patterns

1. We have use Purity as a ranking measures for frequent patterns. It is used to determine the distinctness of frequent pattern in a topic-I file.
2. A phrase is pure in topic t if it is only frequent in titles i.e. topic t and not frequent in documents about other topics.
3. Think of it as sort of a basal noise that you remove from the supports of each pattern in all the topics and you get how much more frequent the pattern is in
4. We have use below formula to calculate the purity of each frequent patterns

purity(p,t)=log [f(t,p) / | D(t) |] - log (max [(f(t,p) + f(t',p)) / | D(t,t') |])

Term →
f(t,p) → frequency of pattern p appearing in topic t.
D(t) → set of documents where there is at least one word being assigned the topic t.
Set D(t) = { d | topic t is assigned to at least one word in document d }
D(t,t') → union of D(t) and D(t').
|- | measures the size of a set.
Actually | D(t) | is exactly the number of lines in topic-i.txt

5. I have used support along with purity in order to distinguish between similar purity valued patterns i.e. two patterns which has same purity value within same topic t are sorted according to their support count.

Arrange by decreasing order of purity
If same purity value then
Sort the same purity value in decreasing order according to support

6. The reason I have used support to distinguish between two same purity valued patterns is that Purity is a term that is un damped because of the negative Logarithm(small float values) while support is not hence mixing them is quite difficult, also purity in itself encompasses support, that was another area of concern for me regarding what kind of mixing should I do so that I don't weigh support twice (duplicate counting)

Source Code →
1. Python File → ***rudani2_apriori.py***
2. Method Name → a. ***func_Purity ()***

Step5: Bonus
1. I have implemented Phraseness and Completeness as filtering/ranking measures.

Phraseness
1. Phraseness is defined as group of words combined together as a phrase if they co-occur significantly more often than the expected chance co-occurrence frequency, given that each term in the phrase occurs independently.

   a. Example: ‘active learning’ is a better phrase than ‘learning classification’ in the Machine Learning topic.
2. I have used probability module to calculate Phraseness. Below is the formula

$$\pi_t^{phr}(p) = \log \frac{P(e_t(p))}{\prod_{w \in p} P(e_t(w))} = \log \frac{f_t(p)}{|D_t|} - \sum_{w \in p} \log \frac{f_t(w)}{|D_t\|}$$

3. Idea is to calculate the probability of pattern ‘p’ and subtract with the summation of probability of all words belongs to that pattern

Observation: -
1. As compared to Frequent Pattern. Phraseness helps to rank bigram and trigram higher than unigram patterns i.e. rank of ['stream', 'data'] > ['stream'] and ['data'] in topic 0 frequent pattern

Source Code →
1. Python File → ***rudani2_apriori.py***
2. Method Name → a. ***func_Phraseness()***

Completeness
1. Completeness is defined as phrase which has no superset i.e. A phrase is not complete if it is a subset of a longer phrase, in the sense that it rarely occurs in a title without the presence of the longer phrase.
   a. Example: ‘support vector machines’ is a complete phrase, whereas ‘vector machines’ is not because ‘vector machines’ is almost always accompanied by ‘support’.
2. I have used superset subset relationship to calculate the completeness.

Observation: -
1. Completeness helps to rank high for the frequent patterns which are superset of some other patterns. i.e ['rule', 'mining', 'association'] is rank higher than ['mining'], ['rule'], ['association'], ['rule', 'association'], ['mining', 'association'] and ['rule', 'mining'] in topic-0 frequent pattern

Source Code →
1. Python File → ***rudani2_apriori.py***
2. Method Name → a. ***func_Completeness()***

Answers to Questions

Question to ponder A*: How do you choose min_sup for this task? Explain how you choose the min_sup in your report. Any reasonable choice will be fine.*

Ans A)
1. I chose the min_support to be 0.01
2. We need to choose min_support in such a way that we balance not thinning down too much (very low support generating too many frequent patterns) as well as not increasing so much so that we cant distinguish between frequent and closed/maximal patterns

3. I tried various support values (0.01,0.002,0.5,0.005 and 0.001) and 0.01 seemed to work best for the given data set
4. 0.01 would mean for a transaction list of 10047 transactions, a pattern is frequent if it appears at least 100 times or more which seems to be a good estimate

Question to ponder B: *Can you figure out which topic corresponds to which domain based on patterns you mine? Write your observations in the report.*

Ans B)

1. Based on the analysis of *.txtphrase files in frequent pattern folder. I was able to decipher the following
    a. Phrase 0 → Associated to Data Mining (DM) Domain
    b. Phrase 1 → Associated to Machine Learning(ML) Domain
    c. Phrase 2 → Associated to Information Retrieval(IR) Domain
    d. Phrase 3 → Associated to Database (DB) Domain
    e. Phrase 4 → Associated to Theory (TH) Domain

Question to ponder C: *Compare the result of frequent patterns, maximal patterns and closed patterns, is the result satisfying? Write down your analysis.*

Ans C)

I have compared the results of Frequent Patterns, Maximal Patterns and Closed Patterns. I have found below observations

1. The results are quite satisfying. The Frequent Patterns in pattern-i.txt files clearly shows all the words that were frequent in the paper titles. Around 75 patterns per topic for min_support 0.01
2. I also found that all of my Closed patterns were also Maximal. I was not able to find non-maximal closed pattern. The reason might be because of comparison between support values. We compare values which are numerically small float values hence checking for equality in small values might have yield incorrect numerical results
3. So only Maximal Patterns are retained which has no superset i.e. (subset → superset)

Question to ponder D*: What are the quality of the phrases that satisfies both min_sup and min_conf? Please compare it to the results of Step1 and put down your observations.*

Ans D)

I have compared the results of Frequent Patterns and Association of Phrases that satisfies both min_sup and min_conf. I have found below observations