

# Hubble Project: User Profiling

Supervisor : Antoine Pigeau  
Co-encadrant : Yannick Prié  
Student : Bukai Lin

# Contents

<b>1</b>	<b>Requirement</b>	<b>3</b>
1.1	A Brief Introduction of The OpenClassrooms . . . . .	3
1.1.1	the Constructure of the Course . . . . .	4
1.1.2	the Introduction of the Premium User . . . . .	4
1.2	the Goal of the Project . . . . .	5
1.3	Input Data . . . . .	6
1.4	Conclusion . . . . .	9
<b>2</b>	<b>Bibliography</b>	<b>10</b>
2.1	the User Profile . . . . .	10
2.2	the Sequences . . . . .	10
2.3	the Library and Package of Frequent Pattern Mining . . . . .	11
2.4	the Fundamental Frequent Pattern Mining Algorithms . . . . .	11
2.4.1	Apriori Method . . . . .	11
2.4.2	FP-growth Method . . . . .	14
2.4.3	Performance Comparison of Apriori and FP-Growth Algorithms . . . . .	17
2.5	Maximal Frequent Itemset . . . . .	18
2.6	Closed Frequent Itemset . . . . .	18
2.7	Sequence Distance . . . . .	18
2.8	Alignment Distance . . . . .	18
2.8.1	Definitions . . . . .	18
2.8.2	Global Alignment . . . . .	19
2.8.3	Local Alignment . . . . .	21
2.8.4	Example . . . . .	24
2.9	Conclusion . . . . .	24
<b>3</b>	<b>Solutions</b>	<b>25</b>
3.1	Generate Sequences . . . . .	25
3.2	Data Preprocessing . . . . .	25
3.3	Data Cleansing . . . . .	26
3.4	Data Transformation . . . . .	26
3.5	Frequent Pattern Mining . . . . .	26
3.5.1	the PrefixSpan Algorithm . . . . .	27
3.5.2	the FPClose Algorithm . . . . .	28
3.6	Conclusion . . . . .	30
<b>4</b>	<b>Experiments</b>	<b>31</b>
4.1	Dataset . . . . .	31
4.2	Pre-processing . . . . .	32
4.3	Data Mining . . . . .	32
4.3.1	Subscription Data . . . . .	32
4.3.2	Reading Session Data . . . . .	33
4.3.3	Premium Data . . . . .	33
4.4	Reading Section . . . . .	35
4.5	Course Data . . . . .	35

4.6	Generate Frequent Sequences . . . . .	36
4.6.1	the Result of FPM . . . . .	36
4.7	Compute Sequence and Alignment Distance . . . . .	38
4.8	Conclusion . . . . .	45

# Chapter 1

## Requirement

In this chapter, we present some basic problems of the user profiling project of the Open Class Room website. The objective is to carry on data mining process according to the user database, in comparing learning traces from the succeed users and failed users in different places, and predicts a new registered user's final mark and classify him/her to the different kind of user.

Firstly, we would do a brief introduction of the main research context and the Open Class Room website. Secondly, we present the goal of this project. Thirdly, we display the fundamental database table to introduce which data we should analysis. Then we will draw a conclusion of this chapter.

### 1.1 A Brief Introduction of The OpenClassrooms

The OpenClassrooms provides online courses open for everyone, there are divers courses included coding, design, business an so on. The registration users can study courses and acquire knowledge by multiply approach such as text, video and ebook. The website also offers some premium services to users, a validation of a course would offer a certificate to the user who chooses the premium when the user pass some exercises and quizzes.

"OpenClassrooms wants to make education accessible by prioritizing a community-based, engaging learning experience." [15] This website no only offer the open courses in accordance with the phases and continuity online to us, but also want to make these education experiments had a positive effect on users' whole professional lives. The OpenClassroom has become the first e-leading site for code learning, tech knowledge, and digital culture in Europe.



Fig.1. the Open Class Room website

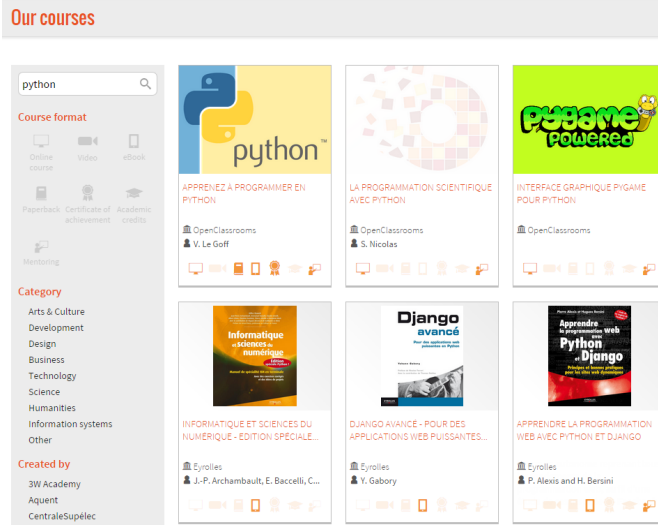


Figure 1.1: the Courses List



Figure 1.2: A Part List in A Course

The online course services open for everyone without age limitation, and all of courses are open, we could use many kinds of devices( PC, iPad, smartphone and so on) to learn online courses anywhere. Users come from all over the world, we could learn courses with various classmates and then we would gain some new idea in this learning environment, some interaction with others in a variety of ways. It provides us with an innovative education in order to make the learning more interest and easy.

### 1.1.1 the Constructure of the Course

The open courses are created by some field experts of teachers, they are available by ebooks (ePub or PDF format) and videos that we could read them on our laptops or other devices. It also provides online learning assistance services, users can ask questions at any time and anywhere. Moreover, users also can learn according to a progress schedule, it's made up of several parts and they are a step by step process, if users follow learning paths, they would have more positive effect on the courses.

In the figures(Figure 1.1 and Figure 1.2), we take python courses as example, now we play a role as a user having a UserId to choose the first python course, put it in detail, we could see the learning structure on this course. This course includes some parts, each part can consist of several session with different position, title and type and they consist of text, images and videos.




There is a quiz at the end of each part and a certificate of the course, then quizzes could be corrected automatically, Moreover, we can find the author's introduction and what is premium service provided at the end of the introduction of the course page.

If you finish all the exercises of the course and the final score is equal to or more than 70, it means you complete the course well and would get the certificates of achievement for the course.

### 1.1.2 the Introduction of the Premium User

The premium user would gain a certificate of a course thanks to some exercises and quizzes, and these certificates prove that we already had enhanced our professional skills in some special field, then they could add into our CVs in order to help us to find our dream job.

There are 3 kinds of premium offers in OpenClassrooms. When the users chooses a premium registration, they can gain more different kinds of services in different premium level. These premium offers divide into premium solo, premium class and premium plus. As the growth of premium level, the course services become more detail and professional, provides more contents and makes the users more competitive. Details are as following graph.

	Free € 0 per month	 Premium Solo € 20 per month	 Premium Class € 90 per month	 Premium Plus € 300 per month
<b>Functionality</b>				
Course	✓	✓	✓	✓
Streaming videos	Limited to 5 per week	Unlimited	Unlimited	Unlimited
Exercises	Limited to 2 per week	Unlimited	Unlimited	Unlimited
Progress dashboard	Limited to 1 course at a time	Unlimited	Unlimited	Unlimited
Course certificates of achievement		✓	✓	✓
HD video downloads		✓	✓	✓
eBook library		✓	✓	✓
Live Lab		✓	✓	✓
Job path achievement certificates			✓	✓
Projects			✓	✓
Dedicated discussion spaces			✓	✓
Group help via mentor			✓	✓
Individual mentor support				✓
State-recognized degree				✓
<b>Financing</b>				
Financing through your business, OPCA (joint commission for collective training)...*		✓	✓	✓
Financing in FPC*				✓

[t]

**Fig.3.** Compare the 3 Premium and Free Offers

Put it in detail, in Premium Solo service: "follow courses at your pace and obtain completion certificates recognized by employers." In Premium class service:"Learn an in-demand skill set wanted by many companies within a small group of motivated students!" In Premium plus service: "Train yourself for a job with a dedicated mentor in order to obtain a title recognized by the State." [15]

## 1.2 the Goal of the Project

This section will discuss about the goal of this project. In this project, we need to find the main reason and mine potential information. Learns the actions which leads to classify the users into success and failed student on the way they browse/learn a course. The goal of this project could be divided into three specific parts:

- Detection of Success/Failed Users

In this specific goal, we need to detect success and failed users based on the following definition. A success student defined as a user who choose a specific course could pass quizzes and then obtain the validation of then course, it also means the success get the mark more than or equal to 70; whereas, it's a failure student.

Firstly, we use the user profiles which are provided from OpenClassrooms, and generate these two kinds of reading sequences for premium users. Secondly, we filter these sequences in order to adapt for data mining,

then analysis some features about the above-mentioned sequences. Moreover, use frequent sequence mining algorithm to find the maximal frequent sequence about success/failed users for each course. Finally, we use sequence distance and alignment distance algorithm to calculate the similarity score between the test user and the each success/failed maximal frequent sequences in the same course, and obtain the alignment sequences about these two sequences, then we decide this test user as success or failed user based on the maximal similarity value.

- Find Class of Users Depending on the Way They Learn/Browse

Each new student register a course, and they would have a reading activities/sessions when they start their courses. These reading sessions have some attributions contains userid, partid, sessionid, duration and so on.

In this section, we would use the pattern mining method to classify the class of users depending on the way they learn or browse. Put it in detail, a set of users have the similar reading activities/sessions, then they would gain the similar score. Therefore, we can predict users' final mark based on different kinds of students' learning trace. If a user who have not finish his/her course yet and have the similar reading session with the previous success users' learning trace, then it's more likely to get a high score and gain a certificate of a course finally. On the contrary, if he/she starts to deviate from the success users' learning trace, maybe this user couldn't pass the course. Moreover, we can classify this two different kinds of users and detect or analysis them. Based on the above-mentioned information, we can predict the users' final learning outcomes assessment according to the way they learn or browse. Therefore, we do the frequent sequence mining to find the maximum frequent sequence, and calculates the alignment score about the test sequence and the previous known sequences.

- Improve the course

To facilitate the quality of course reengineering, we can use reading sessions and associated indicators to realize this goal. [18] And then measuring the efficiency of a learning instrument is also essential in order to revise and improve its quality [12]. Using frequent pattern mining, we can find the frequent sequence for success or failed users' reading trace. When the user who start to deviate from the success trace, in this moment we can let the course told the user adjust their learn trace follow by the success trace, it would make suggestions directly to improve user's learning efficiency and found a better learn method to help the failed student.

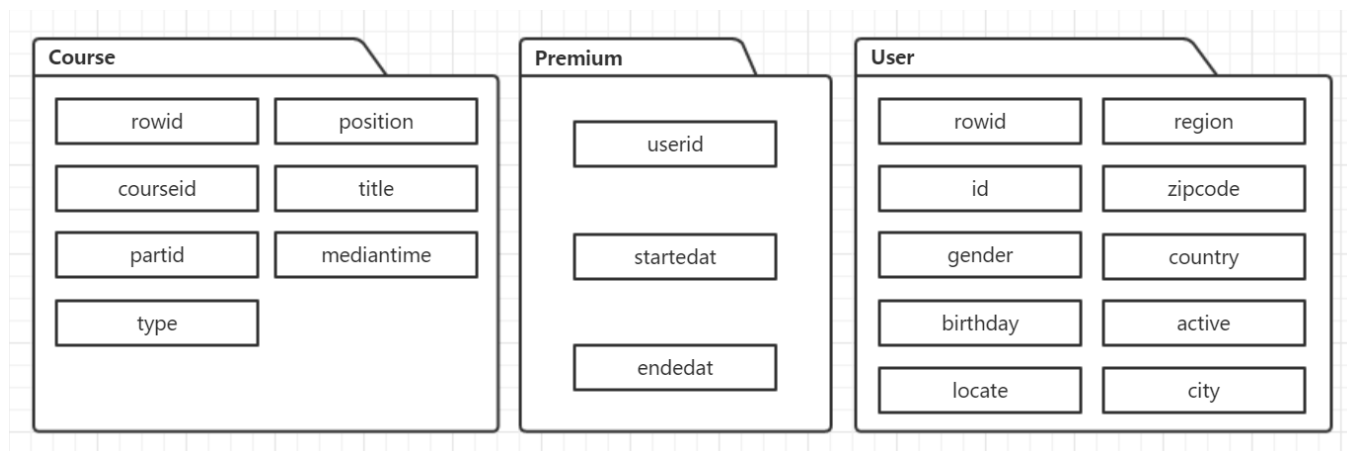
The goal of the project is to built the best suit users' requirements, then update the eLearning document to maximize users' learning efficiency. Makes some suggestions to the teachers and experts who create the courses in order to enhance the quality of courses, put it in detail, these advice include how to adjust the duration of the part in the course, which part should be made users paid more attention and cost more time to learn, find the way to make users more easy to understand some difficulties in the courses and help them to break the bottlenecks.

According the goal of this project, we need to use pattern mining and sequence mining to detect the users' reading behavior and then analysis the success users' learning method, uses frequent pattern mining to find the frequent reading sequence so that improve the quality of the course based on these frequent attributions. Finally, we would make the report and send it to OpenClassrooms team so as to make some suggestion to course designer.

### 1.3 Input Data

There are four different methods about input to use to create the database, respectively, Matrix, Transaction/Sequence of each user on each course and sequence for each user for each course. There are different classification methods in these profile methods, they involved SVM, pattern mining and sequence mining. Finally, we adapted sequence of part for each user on each course as the input method to use to build the profile. Our input data are only based on the database with several tables, From these tables, we would obtain several reading sessions.

The data of this project are implemented and evaluated from OpenClassrooms website and also be constituted from web logs on 4 courses includes java (id:26832), audace d'entreprise (1946116), node-js (1056721) and XML(1766341).



**Fig.4.** the Course, User and Premium Table

Each sequence of actions of a given user includes a lot of information about learning attribute and activity path, we can using sequence mining to reflect and predict users' reading usages and behavior, and classifies different kind of students. According to the sequence of actions of a success user, to makes a better teaching method to help the failure student.

- the Course Table

We create the course table in order to build the course construre, as shown as Fig.2. on the left. We can see there are 7 attributions in this table. Take Audace entreprise course as example, its courseid is 1946116 with 40 partid, these parts can divide into 4 types ranging from small to large type in course, partie, chapter and subchapter located different positions. In this course, it consists of 4 parts, these parts contain 22 chapters, 23 subchapters are included in these chapters,. Each partid type has individual titles with scheduled median time. Such as partid[1946122], there are 5 chapters in this part, they are [L'entrepreneuriat aujourd'hui], [Quest-ce qu'un entrepreneur ?], [Entreprendre: une histoire de vie !], [Ses qualits managriales] and [Apprivoiser le risque] respectively, and then includes two subscriptions, respectively, [Interview de Didier Roche ] and [Interview de Yvon Gattaz].

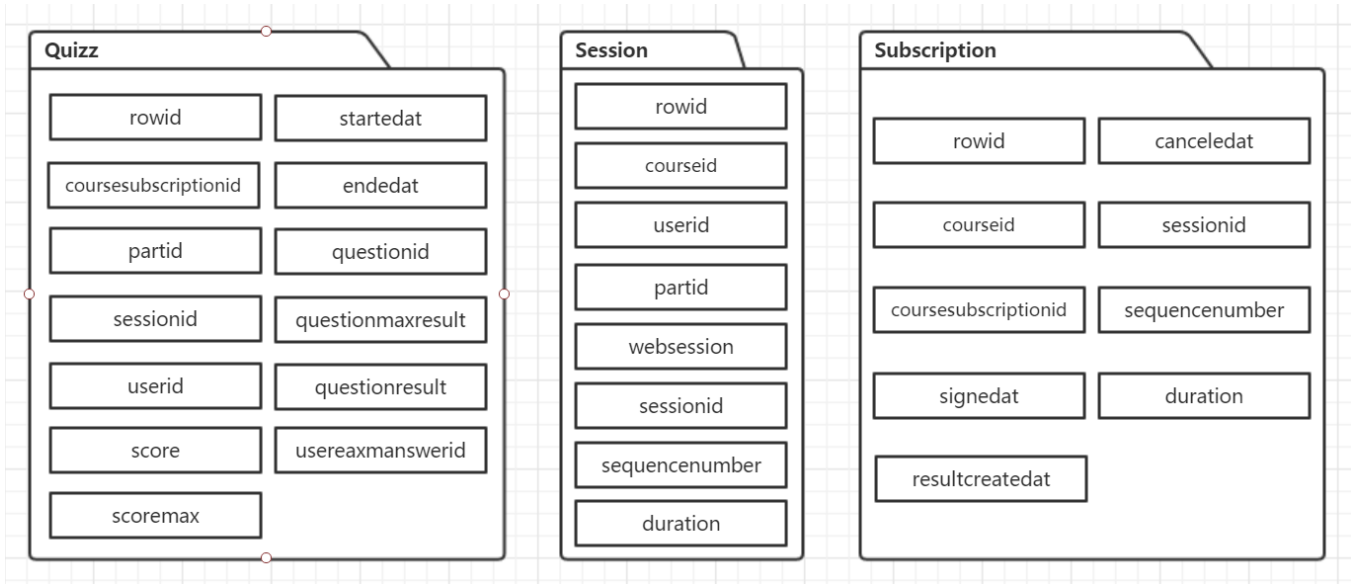
- the User Table

The user table is build for the user profiles as shown as Fig.2 on the right. The attributions in this table includes the fundamental personal information, put it in detail, it contains userid, gender, birthday, locale, region, zipcode, country, active and city. The id is the main, we take useid dFcEDwkJSu6 as an example, selects it from the user table, we can find the rowid is 21622, the user's gender is man, he was born in 1986 and now he is in France and online now. There are several null value in this tuple, but it doesn't affects us to build the user profiles and then analysis them.

- the Premium Table

The premium table is provided by OpenClassrooms as shown as Fig.2 in the middle, it's used to manager the premium users' information. It contains three attributions, respectively, rowid, userid, startedat and endedat. Take the same useid which mentioned in the user table as an example, we can find 2 tuples about this search result. startedat and endedat mean the start time and end time respectively. In this search result we can find this user buy twice premium service with different duration, the first one is from 2015-10-13 to 2016-10-13





**Fig.5.** the Quizz, Session and Subscription Table

and the second one is from 2015-07-08 to 2015-08-08.

- the Quizz Table

The quizz table is the table in order to gather data for quizz, it's as shown as Fig.3. on the left. We also take the same userid as the search key word, we can get 970 results. We can gain each quizz information for each question in any session of any part of any course, we also could know when the user did the question and what's his/her answer and the question result, what is the score he/she got and the maximum value? The above-mentioned information what we can gather are explained the following attributions: rowid, coursesubscriptionid, partid, sessionid, userid, score, scoremax, startedat, endedat, questionid, questionmaxresult, questionresult and usereaxmanserid.

- the Session Table

In eLearning, reading is the most salient learning action and we can understand how learners consume courses through it. [18] The session table is the most important in the database due to it's the basis data of the reading session as shown as Fig.3. in the middle. This table includes 8 attributions: rowid, courseid, userid, partid, websession, sessionid, sequencenumber and duration. The first four attributions had explained in the previous content, so we explain the last four here. websession and sessionid are the same terms, but explain by different kinds. If several tuples have the same sessionid means these reading actions happen at the same time and then it's just a reading session. The sequencenumber is the sum of total number of reading actions, and duration means how lond did reading actions last.

The reading session denotes the period of time during which a reading action occur. It relates to a continuous sequence of reading actions from a user that can be considered continuous (apart from small interruptions). It means that a user actually spends one-hour time on a course will carry out a one-hour reading session. In the concept, the course part reading-time thresholds is a symbol about identifying reading sessions, it's calculated from actual time spent by learners on parts. We can use these thresholds to distinguish reading sessions, so the threshold could be defined as the maximum time needed for reading the corresponding part. [18]

- the Subscription Table

The Subscription Table records the subscription information as shown as Fig.3. on the right. We take courseid [1946116] and userid [dFcEDwkJSu6] as keyword, then we can get 3 record in this table. Each tuple contains

rowid, courseid, coursesubscriptionid, signedat, endedat, canceledat, resultcreatedat, score and userid. For the search result, we can see each action from a user would be inserted in specific row and each subscription of the course also has an id, sign time, end time, result time. When the resultcreatedat isn't null that means a score would appear.

In conclusion, sequence is the most important form and consisted of these input data. Each sequence is a reading session which includes some users' learning attribution. After generating sequences, uses frequent pattern mining to learn these sequences in order to find the different with these two classes.

## 1.4 Conclusion

OpenClassrooms is the first e-leading site in Europe. We want to get the main reason for which factor or learning action lead to divide users into the success and failure. To achieve this object, we would use the previous user profiles and pattern mining to analysis the users' reading behavior in order to detect whether user's learning actions follow the correct path, moreover predict their final result.

In this section, we discuss the method which we adapt to do this project, and talk about which research area we need to study to process the user profiles which we had gathered in the preproduction phase of the project. The data processing method could be divided into three methods as the following introduction.

- Pattern Mining and Sequential Pattern Mining

This method is used to find statistically relevant patterns between data examples where the values are delivered in a sequence. we support the first approach is to search the sequence similarities on a specific course, and the second approach is to search the sequence similarities on a set of courses. This method is the main processing step in the project due to the final result refer to the goal of the project would be produced by this method. There would be more detail introduction in the second chapter.

- Frequent Pattern Mining

Frequent pattern mining is a mature technology which is widely used in the literature, it can solve several data mining or pattern mining problems, for instance clustering and classification. With wide application of Frequent Pattern Mining, its algorithmic aspects have been explored very widely.

To Find the relationships among items in a database is the goal of frequent pattern mining. We assume that there are several transactions  $T_1...T_N$  in a given database D, find out all patterns called P which are appeared in more than once a fraction s of the transactions (where the fraction s is called the minimum support).

Finding the association rules is the begining goal of the original frequent pattern mining, it asks that it should be proposed the closely related to that of frequent patterns. So in general association rules, the second-stage, output which are derived from frequent patterns. All in all, in this technique, find frequent pattern is the first step. In the second step, computational and modeling issues would be arised, especially, when we use it in the context of other data mining problems such as classification. [1] Finally, we will introduce several algorihem about it in the next chapter detailly.

- Clustering

This method is the task of grouping students who have similar action such the traces on browsing or learning courses in the same cluster, and then groups the users in the same cluster into a similar process or similar frequent patterns. Finally we predict the remark of the users according the previous analysis result.

## Chapter 2

# Bibliography

In this chapter, we introduce the concept of sequences which we analysis in this project, explains how we obtain the sequences from the reading sessions in the database and why we want to find the frequent sequences.

Secondly, introduces two fundamental frequent pattern mining algorithm detailly, they are the Apriori Algorithm and the FP-growth algorithm respectively, and do the performance Comparison of Apriori and FP-Growth algorithms, then introduces the algorithm called the PrefixSpan algorithm which we use to do the frequently pattern mining in this project.

Then, introduces the maximal frequent itemset and closed frequent itemset. They would be applied in different solutions in this project.

Finally, we present the distances functions between two sequences. It includes the concept of the sequence distance, global/ local alignment algorithm.

### 2.1 the User Profile

In this section, we introduce the concept of the user profile and discuss how we generate the profiles in the preproduction phase of the project. This user profile would be used to find statistically relevant patterns between data examples where the values are delivered in a sequence. [1]

User profile is the form of personal data associated with a specific user, it could be shown as the digital representation of a person's identity. A profile can store the characteristics of the reading attribution of a user.

Now we discuss the profile of success user and failed user. As the above-mentioned definition in the first chapter, in a word, if a user choose a course and can pass quizzes, obtain the calidation of the course, finally he or she get the mark more than or equal to 70, then he or she is a success user. whereas, it's a failure student.

In practice, we select the premium users where they got marks more than or equal to 70 for the different courses in order to build the profile of success user. Similarly we select the premium and change the selection condition into the mark which is less than 70 for the different courses, then build the profile of premium failed user. Finally, bases on the csv files which are provided from the OpenClassrooms, we can build the user profile for 4 courses.

### 2.2 the Sequences

In this section, we introduce the sequences which are generated in this project. "For each sequence of actions of a given user, the begin time of each of his request is considered as the end time of his previous action to compute the assumed duration of the action." [18]

Firstly, we build the user profile which is composed of reading sessions about 4 courses by two different kinds of users. The sequence of parts is the reading path of a user and it includes the reading sessions. Bases on these

user profile, we generate the two kinds of users' sequences of all the user for 4 courses. Therefore, we would get eight original sequences to analysis, they are JavaSuccess, JavaFailed, AudacEntreprendreSuccess, AudacEntreprendreFailed, NodeJSSuccess, NodeJSFailed, XMLSuccess and XMLFailed respectively.

In these original sequences, they include a list of users who are following for the above-mentioned selecting condition, and idPart, idSession and duration for each user. These three attributions (idPart, idSession and duration) are the specific forms of user reading session. And then we can use these sequences which contain reading attributions to analysis.

## 2.3 the Library and Package of Frequent Pattern Mining

In this section, we introduce SPMF which is an open-source data mining library written in Java, specialized in pattern mining [7], and we use it to process frequently pattern mining. It provides more than 100 data mining algorithms to implementate. In this project, we use the "Mining Frequent Sequential Patterns Using the PrefixSpan Algorithm" in the sequential pattern mining part to carry out the frequent pattern mining.

## 2.4 the Fundamental Frequent Pattern Mining Algorithms

In this section, we discuss two fundamental frequently pattern mining algorithms in order to lead the PrefixSpan Algorithm which we actually use for our project in next section. The following algorithms that we discuss are the support-based framework, in which itemsets with frequency above a given threshold are found.

To process the computational problem is a challenge for frequent pattern mining algorithms, it arouses the high level of interest. Frequent Pattern Mining is widely used in large databases with enormous search space, there are exponential to the length of the transactions in the database. In this project, there are 489285 reading sessions in the database for all users in 4 courses, our goal is that finding the potential attributions in these information, so we use FPM to achieve this goal. Carry out the Frequent Pattern Mining, "the support levels at which one can mine the corresponding itemsets are limited (bounded below) by the memory and computational constraints" [1].

### 2.4.1 Apriori Method

Apriori method is the most basic join-based algorithm, it uses a level-wise approach in which all frequent itemsets of length  $k$  are generated before those of length  $(k+1)$ . [1] The main feature of Apriori method is that if there are some subset in a frequent pattern, so each subset is also frequent. In turn, know frequent patterns of length  $k$  with the use of joins can generate its candidates which length of frequent patterns is  $(k+1)$ . Especially, it's possible that there are two same candidate in candidate generation procedure. In Apriori method, two itemsets would be joined where items in the previous generation are same, and they would be ordered by the lexicographic. Therefore, this method doesn't generate the redundant candidates.

#### The Apriori algorithm

Agrawal and Srikant propose the Apriori algorithm in 1994. This algorithm can operate on databases containing transactions. It's an algorithm which is designed to mine the frequent item, and it's an association rule learning about transactional databases. This algorithm can identify the frequent items and exsert item sets to larger and larger scale until they doesn't exist in the database. [19]

The fundamental Apriori algorithm is run recursively in level-wise fashion and it use a bottom up method. Moreover, there are four steps in the whole algorithm and these steps would be run repeatedly, for different values of  $k$  (where  $k$  is the length of the pattern generated in the current iteration). [9] The detail step is shown as the following

- Uses joins on the patterns in  $Fk$  to generate candidate patterns  $Ck + 1$ .
- The pruning of candidates from  $Ck + 1$ , for which all subsets to not lie in  $Fk$

```

Algorithm Apriori(Database:  $\mathcal{T}$ , Support:  $s$ )
begin
  Generate frequent 1-patterns and 2-patterns
  using specialized counting methods and
  denote by  $\mathcal{F}_1$  and  $\mathcal{F}_2$ ;
   $k := 2$ ;
  while  $\mathcal{F}_k$  is not empty do
    begin
      Generate  $\mathcal{C}_{k+1}$  by using joins on  $\mathcal{F}_k$ ;
      Prune  $\mathcal{C}_{k+1}$  with Apriori subset pruning trick;
      Generate  $\mathcal{F}_{k+1}$  by counting candidates in
       $\mathcal{C}_{k+1}$  with respect to  $\mathcal{T}$  at support  $s$ ;
       $k := k + 1$ ;
    end
  return  $\cup_{i=1}^k \mathcal{F}_i$ ;
end

```

**Fig.6.** The Apriori algorithm

- The validation of the patterns in  $Ck+1$  against the transaction database  $\tau$ , to determine the subset of  $Ck+1$  which is truly frequent.
- The algorithm is terminated, when the set of frequent k-patterns  $Fk$  in a given iteration is empty.

The pseudo-code of the overall procedure is as shown as Fig. 6.

### Example

Now we take a specific example to explain the algorithm, "assume that there is a large supermarket with tracks sales data by stock-keeping unit (SKU) for each item: each item, such as "butter" or "bread", is identified by a numerical SKU. The supermarket has a database of transactions where each transaction is a set of SKUs that were bought together." [3] The following itemsets constitute the transaction database:

Itemsets
1,2,3,4
1,2,4
1,2
2,3,4
2,3
3,4
2,4

Now we carry out the frequent pattern mining using the Apriori algorithm about the above-mentioned database. We set the minimum support value into 3, it means if an itemset is frequent that it would appear more than or equal to 3 transactions in the database.

Calculate the occurrence number of an itemset is the first step of this algorithm, it's defined as the support, of each member item separately, then we scan the database for the first time. We would get the result as following:

Itemsets	Support
1	3
2	6
3	4
4	5

As shown in the table, we can see the support of each itemsets (where their size are equal to 1) is more than or equal to 3, it means all of them are frequent patterns, so we continue to the second step. In the second step, the goal is to generate a list of all pairs of the frequent items:

Itemsets	Support
1,2	3
1,3	1
1,4	2
2,3	3
2,4	4
3,4	3

As shown in the table, those itemsets equal to or exceed the support value which is equal to 3 except the pairs 1,3 and 1,4, it means they satisfy the minimum support, then these itemsets are frequent. Moreover, not only the pairs 1,3 and 1,4 are not frequent, but also any larger set which contains 1,3 or 1,4 cannot be frequent. In this way, we can prune itemsets, and search those frequent triples in the database now, and then exclude all the triples which includes one of these two pairs:

Itemsets	Support
2,3,4	2

In this round, we generate the itemset 2,3,4 and its support value is equal to 2, so it's not a frequent pattern due to it's less than the minimal threshold. Moreover, its all super sets are also not frequent pattern, then we don't need to continue to run the next round. Finally, We have thus determined the frequent sets of items in the database, and illustrated how some items were not counted because one of their subsets was already known to be below the threshold. [3]

### Limitations

The Apriori algorithm attempts to load up the candidate set with as many as possible before each scan. Bottom-up subset exploration (essentially a breadth-first traversal of the subset lattice) finds any maximal subset S only after all  $2^{|S|} - 1$  of its proper subsets. [19]

In fact, the Apriori algorithm has flaws and limitations. Put it in detail, there are two aspect limitations about it as shown as following [17]:

- A huge number of candidate sets  
For instance, there are  $10^4$  frequent 1-itemsets, if we use Apriori algorithm to find the frequent itemset, we would get more than  $10^7$  candidates with length-2 and then we need to accumulate and calculate their occurrence number, therefore, it's a time-consuming algorithm. Furthermore, if we mine a frequent pattern which size is 100, and its form like  $(a_1, \dots, a_{100})$ , there are  $2^{100}$  to  $2^{10^{30}}$  candidates generated finally. Therefore, we cannot find an implementation technique to achieve it.

- Scan database repeatedly

Using this algorithm to mine the frequent pattern, it would scan the database again and again until these conditions evaluate to false, the whole loop terminates. Then we still need to check a numerous set of candidates in order to match the pattern, which is true for mining long patterns.

## 2.4.2 FP-growth Method

The FP-growth algorithm is currently One of the currently fastest and most popular approaches to frequent item set mining. This algorithm is based on a prefix tree representation of the given database of transactions and it's called an FP-tree, which can save considerable amounts of memory for storing the transactions. [5]

### The FP-growth algorithm

The FP-growth approach combines suffix-based pattern exploration with a compressed representation of the projected database for more efficient counting. The prefix-based FP-Tree is a compressed representation of the database which is built by considering a fixed order among the items in an itemset. [1]

Now we introduce the FP-growth algorithm detailly. Firstly, count occurrence number of itemsets (attribute-value pairs) in the database, then stores them to 'header table'.

Secondly, based on the inserting instances, we build the FP-tree structure. Classify the itemsets of each instance in frequency descending order in the database, in order that process the tree fastly. We just keep the itemset which is more than or equal to the minimum coverage threshold. And the FP-tree provides high compression close to tree root when many instances share most frequent items.

This algorithm generates the larger and larger itemsets by carrying out the compressed version of main database resurively and directly, avoid to grow and check the candidates aganist the total database. Growth starts from the bottom of the header table (having longest branches), Find all instances which satisfies the given condition. And then the new tree would be created, with counts projected from the original tree corresponding to the set of instances that are conditional on the attribute, with each node getting sum of its children counts. The recursive growth would be cease when any itemsets couldn't satisfy the minimum support threshold, and it means that we cannot find more frequent pattern in this round. Then continue to carry out the remaining header items of the original FP-tree. [11]

"Once the recursive process has completed, all large item sets with minimum coverage have been found, and association rule creation begins." [21]

The pseudo-code of the FP-growth algorithm is as shown as Fig.7.

### Example

In order to understand how the FP=growth algorithm run, now we take an example to explain it. The transaction table is shown as the following:

TID	List of item IDs
T100	I1,I2,I3
T200	I2,I4
T300	I2,I4
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

**Algorithm** *FP-growth*(FP-Tree on Frequent Items: *FPT*,  
Minimum Support; *s*, Current Itemset Suffix: *P*)

```

begin
  if FPT is a single path or empty
    for each combination C of nodes in path do
      report all patterns  $C \cup P$ ;
    else
      for each item i in FPT do
        begin
          Generate pattern  $P_i = \{i\} \cup P$ ;
          report pattern  $P_i$  as frequent;
          Use pointer-chasing to extract conditional
            prefix paths for item i;
          Construct conditional FP-Tree  $FPT_i$  from conditional
            prefix paths after removing infrequent items;
          if ( $FPT_i \neq \phi$ ) FP-growth( $FPT_i$ ,  $P_i$ , s)
        end
      end
    end

```

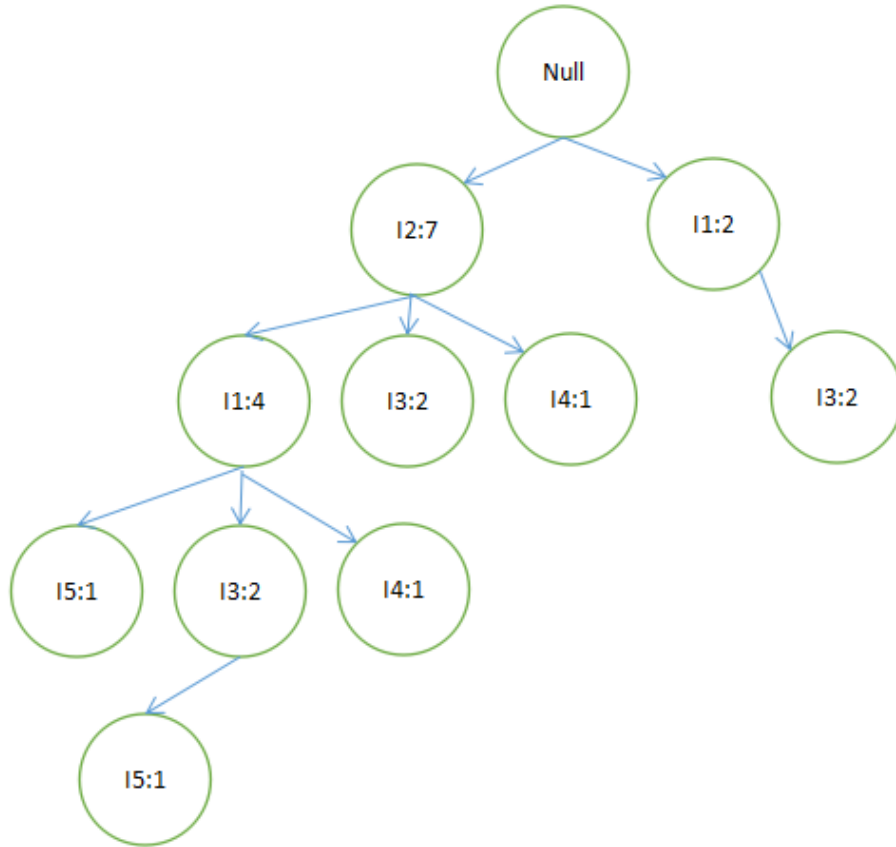
**Fig.7.** The FP-Growth algorithm

Now we create a FP-tree related to this transaction table, and the itemsets are sort by the order of their descending value of support count.

Item	occurrence number
I2	7
I1	6
I3	6
I4	2
I5	2

To facilitate tree traversal, an item header table is built in order that each item points to its occurrences in the tree via a chain of node-links. It's as shown as the figure 8.





**Fig.8.** FP Tree Construction

After the previous procedure which related to FP Tree construction, we need to find conditional pattern base and conditional FP Tree for each item. The result is shown as the following table

	<b>Frequent Pattern Generated</b>
I5	(I1,I2:2),(I1,I5:2),(I2,I1,I5:2)
I4	(I2,I4:2)
I3	(I2,I3:4),(I1,I3:4),(I2,I1,I3:2)
I1	(I2,I1:4)

### Advantages and Disadvantages of FP-Growth Algorithm

- Advantages

The FP-Growth algorithm scans the database only twice and the database seem to be compressed, then it wouldn't generate any candidate, therefore, it run much faster than the Apriori algorithm.

- Disadvantages

The FP-tree may not fit in memory is the first defect; the second one is that it is too expensive to build. Moreover, the frequent itemsets would be found easily, although we need to take time to build it. It would become time-consuming when the support threshold is high, because the only pruning that can be done is on single items.

We can see the relationship between run time and support threshold about the FP-Growth and Apriori algorithm in figure 9

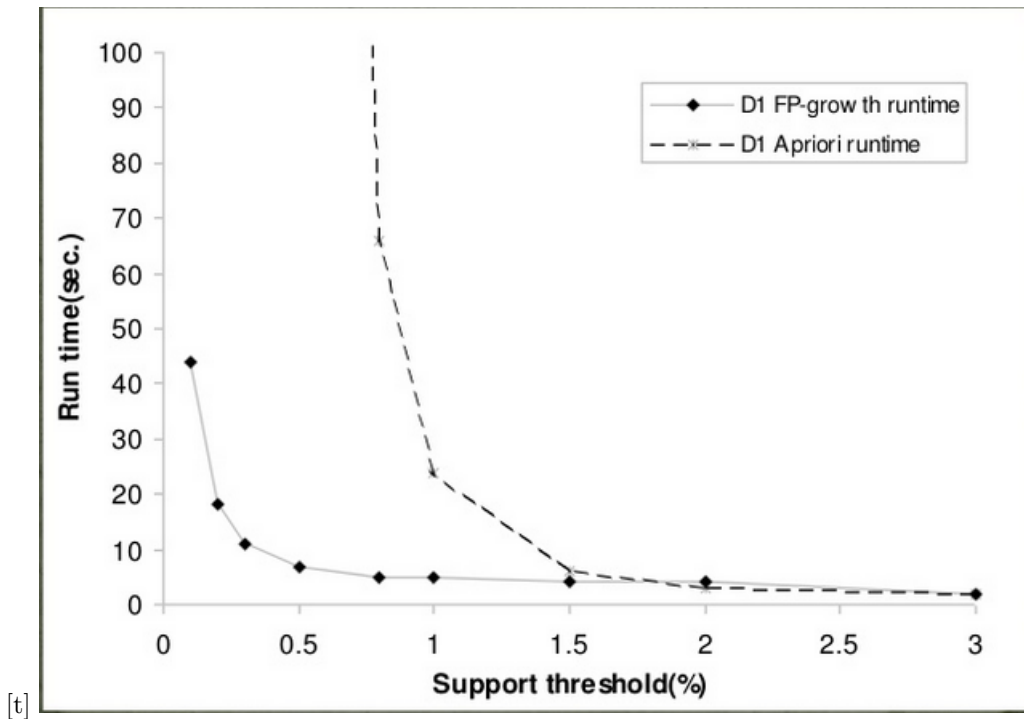


Fig.9. Comparison with the FP-Growth and Apriori algorithm

### 2.4.3 Performance Comparison of Apriori and FP-Growth Algorithms

The candidate generating algorithms (derived from Apriori) behave well only for small databases (max. 50,000 transactions) with a large support factor (at least 30%). In other cases the algorithms without candidate generation FP-growth behave much better. [9]

Apriori and FP-Growth Algorithms belong to the Association Rule Mining about interesting pattern mining problem. This two algorithms used are conceptually clear and ensuing results are perceivable. From the experimental data conferred, it is concluded that the FP-growth algorithm performs better than the Apriori algorithm. In future, it is possible to extend the research by using the different clustering techniques and also the Association Rule Mining for large number of databases. [14]

[t]

Algorithm	Technique	Runtime	Memory usage	Parallelizability
<b>Apriori</b>	Generate singletons, pairs, triplets, etc.	Candidate generation is extremely slow. Runtime increases exponentially depending on the number of different items.	Saves singletons, pairs, triplets, etc.	Candidate generation is very parallelizable
<b>FP-Growth</b>	Insert sorted items by frequency into a pattern tree	Runtime increases linearly, depending on the number of transactions and items	Stores a compact version of the database.	Data are very inter dependent, each node needs the root.

Fig.10. Apriori vs FP-Growth

## 2.5 Maximal Frequent Itemset

If a frequent itemset  $P \in F$  doesn't have frequent superset, then it's called maximal. Let  $M$  be the set of all frequent maximal itemsets, which can be written as the following:

$$M = \{P | P \in F \text{ and } \nexists Q \supset P, \text{ such that } Q \in F \}$$

## 2.6 Closed Frequent Itemset

A closed itemset is defined as a largest element concerning the number of items in each equivalence class. if  $P$  is a frequent itemset, it's a closed frequent itemset where  $\gamma(P) = p$ . ( $\gamma$  is the closure operator). If there isn't any frequent superset with the same support, it means the itemset is closed. A closed itemset  $C$  could be written as the following: [1]

$$C = \{P | P \in F \text{ and } \nexists Q \supset P, \text{ such that } \text{support}(Q) = \text{support}(P) \}$$

## 2.7 Sequence Distance

Sequence distance is the topic about measure the distance between two sequence, put it in detail. Suppose we have two sequences  $X$  and  $Y$ , where  $X = x_1, x_2, \dots, x_m$  and  $Y = y_1, y_2, \dots, y_n$  respectively. The point is how similar these two sequences are?

There are many methods to obtain the sequence distance such as the practical measure ( the score using the sum of the pairwise scores), the nice metric, the dynamic programming and the maximum bipartite matching. In this project, we calculate the sequence distance between the test sequence from a new user and the previous sequences from success or failed users. The similarity is the important factor to predict that the new user would be a success or failed user.

Take the Audace Entrepreneurs course as the example, there is a new user having the reading session, then we process and filter this reading sequence in order to adapt the sequence distance operation. Firstly, we calculate the sequence distance between the test sequence and each sequence from success users, and get the average similarity value as the result. Secondly, we calculate the sequence distance between the test sequence and each sequence from failed users, and also get the average value. Finally, if the value calculated with success users is bigger than failed users, it means this new user's reading sequence is more similar to the success and he/she is more likely to achieve success.

In the particular case, we use the first Audace Entrepreneurs success sequence as the test sequence [1946136 1946141 1946176 1946136 1946176 1946181 1946181 1946181 1946191 1946251 1946251 1946291 1946336 1946426 1946431 1946431 1946451], and the second one is [1946136 1946141 1946176 1946136 1946176 1946181 1946181 1946181 1946191 1946251 1946251 1946291 1946336 1946426 1946431 1946451 1946426]. Run these two sequence in Python platform using difflib package, and the sequence distance value is 0.882352941176.

## 2.8 Alignment Distance

Sequence alignment falls into a set of problems with a long history in computer science, and the underlying metric for distances between sequences falls in the province of biology. It assess similarity of sequences and learn about their evolutionary relationship, we can see it detailedly as the following example:

### 2.8.1 Definitions

Before the definitions about the sequence alignment, we introduce the dynamic programming and it is applied to any number of sequences in theory. In This technique, it requires constructing two sequences with same dimensional equivalent of the sequence matrix formed. A normal dynamic programming is used on all pairs of query sequences and then the "alignment space" is filled in possible gaps or matches at intermediate positions, finally building an

alignment essentially between each two-sequence alignment. [13] dynamic programming are used widely in many several computer science areas, and Needleman-Wunsch and Smith-Waterman algorithms for sequence alignment are defined by dynamic programming approach. Now we assume two sequences called S and T in order to perform alignments, set the length of S and T into n and m, the i-th position of S is  $s_i$  and j-th position of T is  $t_j$ .

## 2.8.2 Global Alignment

A global pairwise alignment A of S and T results from inserting gaps into both S and T such that the two augmented sequences, after the insertion of gaps, are the same length. [6] It carries out an end-to-end alignment of the query sequence with the reference sequence, and global alignment technique focuses on closely related sequences of similar lengths, The normal global alignment technique is based on the Needleman-Wunsch algorithm which is a dynamic programming technique. [16]

The NeedlemanWunsch algorithm is a useful algorithm for the global alignment, it's usually used to align sequences and compare sequences. [4] It can divide a large problem into a set of smaller problems and then solve these smaller problems in order to process the larger problem. Take a specific example to explain this algorithm now, it's shown in the following two sequences:

CGTGAATTCAT and GACTTAC

The length of these two sequences are 11 and 7 respectively. The initial matrix includes (A+1) columns and (B+1) rows (where A = 11 and B = 7). The given extra row and column is used to align with gap and the initial matrix is shown in Figure 11.

-	C	G	T	G	A	A	T	T	C	A	T
G											
A											
C											
T											
T											
A											
C											

Fig.11. Initial matrix

Now we introduce the score system. The basic score system is assumed as, if two element at  $i^{th}$  and  $j^{th}$  position are same value, the score is equal to 1 and it's called matching score. Additionally, if two element at  $i^{th}$  and  $j^{th}$  position are different, then score is equal to -1 and it's called mismatching score. Moreover, in the gap score or gap penalty case, the score is also equal to -1.

There are 3 steps in this algorithm.

- Initialization Step

There is gap penalty in the example, so the first column and row would be filled in 0, at the same time, the gap score would be added to the previous cell of the column or row as shown as the Figure 12.

	-	C	G	T	G	A	A	T	T	C	A	T
-	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11
G	-1											
A	-2											
C	-3											
T	-4											
T	-5											
A	-6											
C	-7											

Fig.12. Initialization of matrix

- Matrix Fill Step

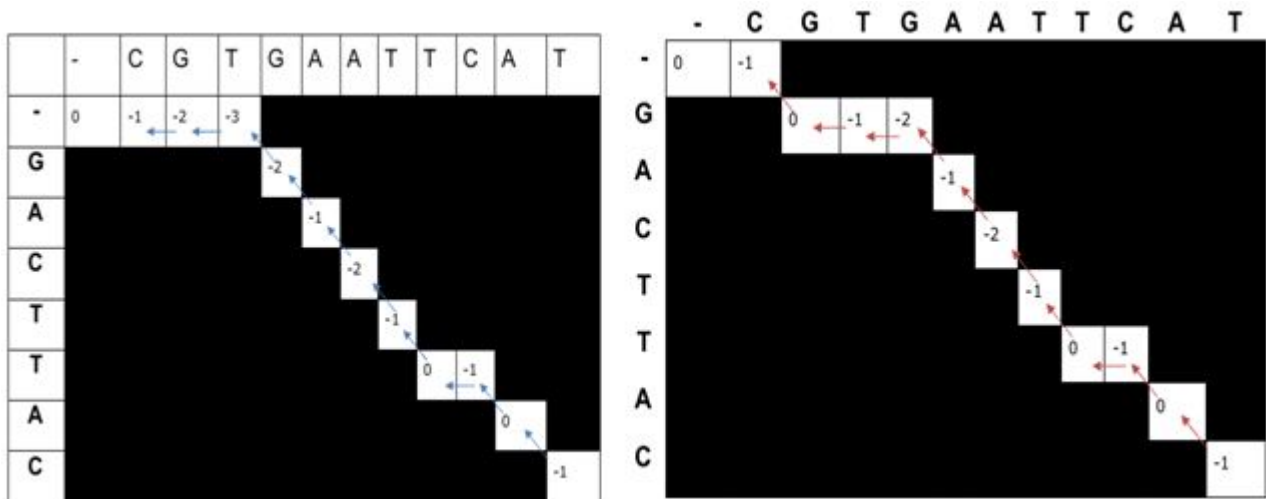
This step is crucial for the algorithm, we would fill the matrix from the upper left hand corner of this matrix. We need to find the maximum score of each cell in order to fill the matrix, before this operation, knows the neighbouring scores of the current position (from left, right and diagonal). From the assumed values, we add the different score to the left, right and diagonal values, therefore, there are 3 different values which we got. And then we take the maximum value among these values and fill the  $i^{th}$  and  $j^{th}$  position with it. In position (1,1) of the matrix, the score is equal to -1, we fill all the remaining grid as same as the position (1,1), then the matrix is shown as figure 13.

	-	C	G	T	G	A	A	T	T	C	A	T
-	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11
G	-1	-1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9
A	-2	-2	-1	-1	-2	-1	-2	-3	-4	-5	-6	-7
C	-3	-1	-2	-2	-2	-2	-2	-3	-4	-3	-4	-5
T	-4	-2	-2	-1	-2	-3	-3	-1	-2	-3	-4	-3
T	-5	-3	-3	-1	-2	-3	-4	-2	0	-1	-2	-3
A	-6	-4	-4	-2	-2	-1	-2	-3	-1	-1	0	-1
C	-7	-5	-5	-3	-3	-2	-2	-3	-2	0	-1	-1

Fig.13. Matrix filling with back pointers

- Trace back Step

Finally, the score is equal to -1 in the bottom right hand corner for this example. There may be more than one alignments possible between those two sequences. Obviously, this cell is equal to -1, it has the maximum score obtained is diagonally located and it's equal to 0. There may be more than one values which points back, it means there can be more than one possible alignments. Carry out the trace back step until the  $0^{th}$  column and  $0^{th}$  row. The alignment of these two sequences could be found when we finish the previous steps.



**Fig.14.** The possible Alignments with trace backing

### 2.8.3 Local Alignment

A local alignment  $A$  of  $S$  and  $T$  is a global alignment of a consecutive substring of symbols of  $S$  with a consecutive substring of  $T$ . These will form the basis of the bulk of this survey. [6] In case of the dissimilar sequences, local alignments technique is more useful and it's know as that it includes regions of similarity or similar sequence motifs within their larger sequence context. The normal local alignment technique is based on the Smith-Waterman algorithm which is a dynamic programming technique.

The Smith-Waterman Algorithm is used to determine similar regions between two sequences. It doesn't need to check the total sequence, this algorithm compares segments of all possible lengths and optimizes the similarity measure. It mainly differs within two aspects from the Needleman-Wunsch algorithm. Firstly, local alignment differs by having only a negative score for the mismatch and when the matrix value becomes negative it has to be set to zero, which renders the (thus positively scoring) local alignments visible. [2]

In the following section, we talk about the assumed scoring schemas: If the element are same in both the sequences the match score is equal to  $+5$  which is added to the diagonally positioned cell of the current cell ( $i, j$  position). If it's different, the mismatch score is equal to  $-3$ . And the gap penalty score is equal to  $-4$  which is added to left and above positioned cells of the current cell. Actually, these scores aren't always unique, moreover the mismatch score and gap penalty should be the negative values. Now Take a specific example to explain this algorithm now, it's shown in the following two sequences:

CGTGAATTCAT and GACTTAC

There are 3 steps in this algorithm.

- Intialization of Matrix

The initial matrix is created with  $(A+1)$  columns and  $(B+1)$  rows ( the length of these two sequences are 11 and 7 respectively, therefore, where  $A = 11$  and  $B = 7$ ). The values in the first row and first column are set to zero as shown in Figure 15.

	-	C	G	T	G	A	A	T	T	C	A	T
-	0	0	0	0	0	0	0	0	0	0	0	0
G	0											
A	0											
C	0											
T	0											
T	0											
A	0											
C	0											

Fig.15. Initialization of Matrix

- Matrix Fill Step

As same as the NeedlemanWunsch algorithm and based the Smith-Waterman assumed scoring schemas, after the procedure about filling, keep the pointer back to the cell from where the maximum score has been determined. The filling matrix would be shown in Figure 16.

	-	C	G	T	G	A	A	T	T	C	A	T
-	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	5	1	5	1	0	0	0	0	0	0
A	0	0	1	2	1	10	6	2	0	0	5	1
C	0	5	1	0	0	6	7	3	0	5	1	2
T	0	1	2	6	2	2	3	12	8	4	2	6
T	0	0	0	7	3	0	0	8	17	13	9	7
A	0	0	0	3	4	8	5	4	13	14	18	14
C	0	5	1	0	0	4	5	2	9	18	14	15

Fig.16. Matrix filling with back pointers

Each cell is back pointed by one or more pointers from where the maximum score has been obtained. [2]

- Trace backing the sequences for an optimal alignmen

Before the final step is trace backing, we need to find out the maximum score in the matrix for the local alignment of the sequences. There may are more than one cell that they obtain the maximum scores, it means there may be two or more alignments, and the best alignment by scoring it. In this case, the maximum score is 18 in the matrix, it appear twice that lead to multiple alignments, so the best alignment has to be found. Finally, the trace back process starts from the highest value, pointing back with the pointers, and search the possible predecessor, then move to next predecessor and continue until the score 0. The result is shown as the figure 17.

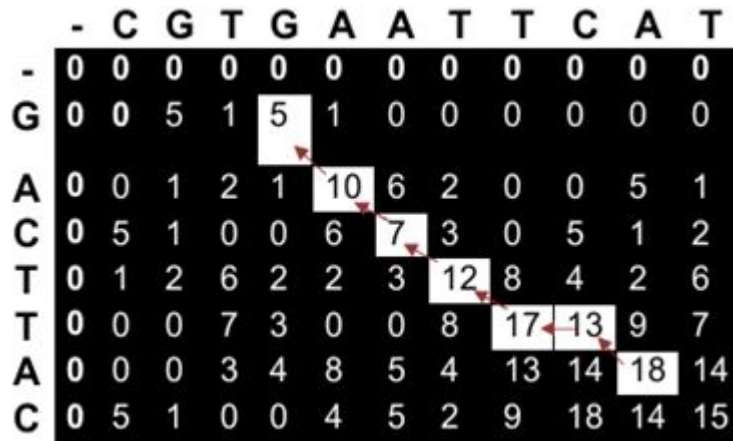


Fig.17. Trace back of first possible alignment

We can find the second pointers pointing our from the maximum cell, it's shown as the figure 18 as the following.

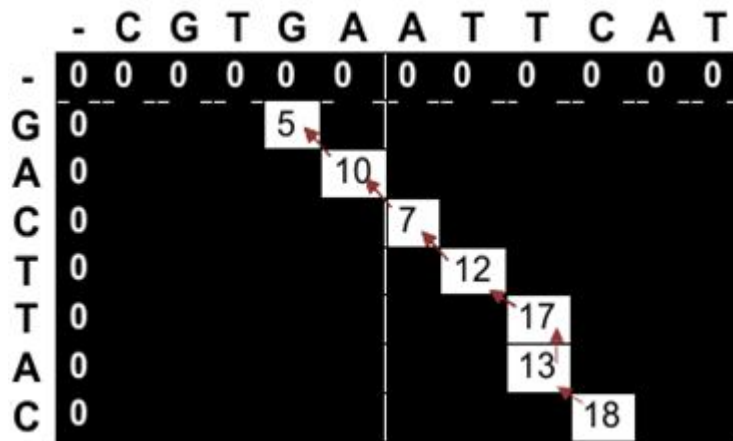


Fig.18. Trace back of second possible alignment

Finally, we would obtain the local alignment and the possible alignments as in Figure 19.

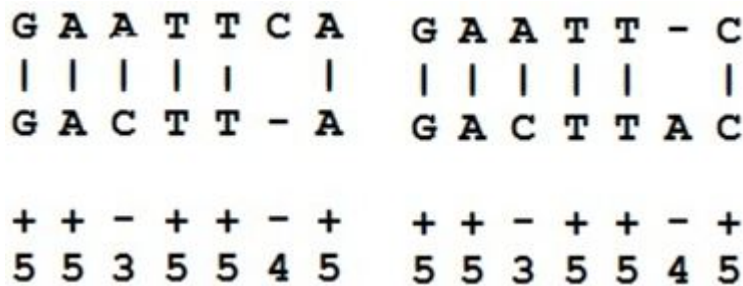


Fig.19. Scoring for best alignment

we summed up the scores both of the alignments are giving the same as 18, so one can predict both alignments are the best.



## 2.8.4 Example

Take a specific example to explain this concept. In the Audace Entrepreneur course, there are 5 sequences for success users. We use the first sequence [1946136 1946141 1946176 1946136 1946176 1946181 1946181 1946181 1946191 1946251 1946251 1946291 1946336 1946426 1946431 1946431 1946451] as the test sequence in order to calculate the alignment score and percent identity with the others. The second success sequence is [1946136 1946141 1946176 1946136 1946176 1946181 1946181 1946181 1946191 1946251 1946251 1946291 1946336 1946426 1946431 1946451 1946426], so we use these two sequences to be aligned. Run the code for them in python platform, the alignment score is 30 and the percent identity is 94.1176470588.

Then these two sequences would be aligned as the following form:

```
1946136 1946141 1946176 1946136 1946176 1946181 1946181 1946181 1946191 1946251 1946251 1946291 1946336
1946426 1946431 1946431 1946451
```

```
1946136 1946141 1946176 1946136 1946176 1946181 1946181 1946181 1946191 1946251 1946251 1946291 1946336
1946426 - 1946431 1946451
```

## 2.9 Conclusion

Study and mine the user profile is the initial step, it could show us the research object directly. Sequence is the fundamental unit in this project, we generate several sequences from the user profile and then we process frequent pattern mining based on these sequences in order to classify the different user.

Frequent pattern mining algorithm is the basic knowledge which we need to learn before carrying out the project, therefore, introduce two fundamental algorithm and one actually used algorithm in this chapter, and compare their con and pon.

In order to classify a user into the known class, we use sequence distance and alignment distance knowledges to process it.

# Chapter 3

## Solutions

In this chapter, firstly, we would discuss the solutions about how to generate the specific sequences and how to obtain the frequent sequences for each courses using frequent patter mining algorithm, and how many stages we process. Secondly, we would talk about which algorithm we used in the project, why we choose it, and what is the advantage or drawback of this method.

### 3.1 Generate Sequences

Bases on the initial data from Openclassrooms, we built the database in order to process pattern mining or frequency pattern mining. These data include the structure of courses, reading session, results of assessment and quizz, subscription for each courses, user profiles and premium users csv files. Due to analysis the potential reading attributes and improve the learn efficiency, we define several fundamental tables which include the basis information that we need to analysis, uses the database software called SQLite. So we created six fundamental table, respectively, course, quizz, session, subscription, trace and user.

At first, we generate the sequences of all the user for the specific course based on the course id, then return a list which form is such as [(idUser, [(idPart, idSession, duration), (), ...,())], ...]. These lists include the reading sessions from all the users, this original reading session include idPart, idSession and duration, and these three feature were sorted based on the section and ordered by idPart.

Especially, this project focuses on the premium users' information. Therefore, we get two kinds of sequences of the premium users for a specific course, success sequences and failed sequences are devided by the mark which is equal to 70. Then the form of sequences is as same as the above-mentioned form.

### 3.2 Data Preprocessing

Before using the data mining algorithm, we need to assemble the original sequences which are generate in the previous step. Filter the sequences is indispensable due to pre-processing is essential to analyze the multivariate data sets before data mining. "As data mining can only uncover patterns actually present in the data, the target data set must be large enough to contain these patterns while remaining concise enough to be mined within an acceptable time limit." [20]

The original form of sequence is shown as:

```
(Courseid,[(User1,[(Part1,Session1,duration1),(Part2,Session2,duration2,...,(PartN,SessionN,durationN)]),
(User2,[(Part1,Session1,duration1),(Part2,Session2,duration2,...,(PartM,SessionM,durationM)]),...,
(UserN,[(Part1,Session1,duration1),(Part2,Session2,duration2,...,(PartR,SessionR,durationR)]))])
```

Due to carry out sequence mining based on partid, we need to filter the original sequence in order to make it suitable for this requirement. Therefore, I create filterSequence function in dataProcessing class to finish this operation, and its input data is from getSequencesSucceedPremium function. It can get the different arguments of the users for a specific course and remove the idPart belong to idCourse or not, and return several different list for

different mining requirement and specific form is shown as:

```
[(idPart, idPart, ... ,idPart),(idPart, idPart, ... ,idPart),..., (idPart, idPart, ... ,idPart)]
[(idSession, idSession, ... ,idSession),(idSession, idSession, ... ,idSession),..., (idSession, idSession, ... ,idSession)]
[(duration, duration, ... ,duration),(duration, duration, ... ,duration),(),..., (duration, duration, ... ,duration)]
[((idPart, duration),(idPart, duration)...),(((),()...()))...((idPart, duration),(idPart, duration)...)]
[(idUser,((idPart,duration),(idPart,duration)...)),..., (idUser,((idPart,duration),(idPart,duration)...)))]
```

### 3.3 Data Cleansing

The goal of data cleansing is to modify, detect and remove those corrupt or erroneous records from itemsets, list, table or database. "Used mainly in databases, the term refers to identifying incomplete, incorrect, inaccurate, irrelevant, etc. parts of the data and then replacing, modifying, or deleting this dirty data or coarse data." [20]

The filter sequences still have reduplication idPart. It means the idPart appear more than once in the same sessionid, this idPart is redundance, therefore, we need to remove it and add the duration value of the same idPart in the same sessionid. The prepareInputDataPForCourse function in dataProcessing class can output a new non-redundance idPart list as ((idPart,idPart,...idPart),(idPart,idPart,...idPart),..., (idPart,idPart,...idPart)) group by session id.

As the same time, the idPart keep the order by sessionid, and this operation could be done by removeReduplicatePart function which to output a re-organise list without reduplicate idPart in the same idSession and keeps the original order, and RemoveSession function which can remove the idSession attributions in order to satisfy the input type for Frequent Pattern Mining operation. The GroupSession function. All the previous mentioned function are in dataProcessing class.

### 3.4 Data Transformation

In data mining technique, the goal of data transformation is to transfer a set of data from the original format into a specific format which adapts to the algorithm. Normally, there are two steps in data transformation procedure as shown as the following: [20]

- Data mapping match data elements from the original format to the format which the destination system requires.
- code generation that creates the actual transformation program

In this project, we got the sequences generated by the InputDataPGroupBySession function which had mentioned in the last chapter. Now we need to carry out data transformation in order to be suitable for the frequent pattern mining operation. Put it in detail, we use the exportSequenceToTXTFile function to export a sequence into a txt file with specific format. There are two way to generate the frequent sequences in this project. Therefore, we would generate two different kinds of data format to adapt the next operation.

### 3.5 Frequent Pattern Mining

The Apriori algorithm still encounters problems when a sequence database is large and/or when sequential patterns to be mined are numerous and/or long. [10]. In this project, there are several user profiles need to be mined. Therefore, we adapt two methods to find the frequent sequences. The first one uses the PrefixSpan Algorithm to

mine frequent sequential patterns, and then get the maximal frequent itemset for the results. The second one use FPClose algorithm to mine frequent closed itemsets.

### 3.5.1 the PrefixSpan Algorithm

The PrefixSpan algorithm is defined as Prefix-projected Sequential pattern mining and it explores prefix-projection in sequential pattern mining. The projecting sequence databases doesn't consider all the possible occurrences of frequent subsequences, it just focus on frequent prefixes due to ant frequent subsequence could be found by growing a frequent prefix.

Before we introduce the algorithm, some parameters should be denoted by the following:

$\alpha$  : a sequential pattern;  $l$  : the length of  $\alpha$ ;  $S|\alpha$  : the  $\alpha$  - projected database, if  $\alpha \neq \langle \rangle$ ; otherwise, the sequence database  $S$ .

There are three steps in this algorithm [10]:

- S, Find the set of frequent items  $b$  by scanning the  $\alpha$  - projected database once, such as the following format:
  - The frequent items  $b$  could be configured to the last element, so that to constitute a sequential pattern.
  - $\langle b \rangle$  could be supplemented to  $\alpha$  in order to constitute a sequential pattern.
- Supplement each frequent item  $b$  to  $\alpha$  to constitute a sequential pattern  $\alpha'$  and then output it.
- Construct  $\alpha'$ -projected database  $S|\alpha'$  for each  $\alpha'$ , finally call PrefixSpan ( $\alpha', l + 1, S|\alpha'$ )

#### Example

We take the sequence database S in the following table and where the minimum support is equal to 2.

Sequence ID	Sequence
10	(a(abc)(ac)d(cf))
20	((ad)c(bc)(ae))
30	(ef)(ab)(df)cb
40	(eg(af)cbc)

- Find length-1 sequential pattern  
 The first step is to find length-1 sequential patterns [10], this step scans the sequence database once to mine all frequent items and every frequent item is a length-1 sequential pattern. In this example, They are  $\langle a \rangle: 4$ ,  $\langle b \rangle: 4$ ,  $\langle c \rangle: 4$ ,  $\langle d \rangle: 3$ ,  $\langle e \rangle: 3$  and  $\langle f \rangle: 3$
- Divide search space  
 In this step, we can partition the complete set of sequential patterns into the above-mentioned six subsets
- Find subsets of sequential patterns  
 The constructing related projected databases could mine the subsets of sequential patterns and then mine each recursively [10]. We can see the projected database and sequential patterns in the following table.

Prefix	Projected (postfix) database	Sequential patterns
$\langle a \rangle$	$\langle (abc)(ac)d(cf) \rangle, \langle (-d)c(bc)(ae) \rangle, \langle (-b)(df)cb \rangle, \langle (-f)cbc \rangle$	$\langle a \rangle, \langle aa \rangle, \langle ab \rangle, \langle a(bc) \rangle, \langle a(bc)a \rangle, \langle aba \rangle, \langle abc \rangle, \langle (ab) \rangle, \langle (ab)c \rangle, \langle (ab)d \rangle, \langle (ab)f \rangle, \langle (ab)dc \rangle, \langle ac \rangle, \langle aca \rangle, \langle acb \rangle, \langle acc \rangle, \langle ad \rangle, \langle adc \rangle, \langle af \rangle$
$\langle b \rangle$	$\langle (-c)(ac)d(cf) \rangle, \langle (-c)(ae) \rangle, \langle (df)cb \rangle, \langle c \rangle$	$\langle b \rangle, \langle ba \rangle, \langle bc \rangle, \langle (bc) \rangle, \langle (bc)a \rangle, \langle bd \rangle, \langle bdc \rangle, \langle bf \rangle$
$\langle c \rangle$	$\langle (ac)d(cf) \rangle, \langle (bc)(ae) \rangle, \langle b \rangle, \langle bc \rangle$	$\langle c \rangle, \langle ca \rangle, \langle cb \rangle, \langle cc \rangle$
$\langle d \rangle$	$\langle (cf) \rangle, \langle c(bc)(ae) \rangle, \langle (-f)cb \rangle$	$\langle d \rangle, \langle db \rangle, \langle dc \rangle, \langle dcb \rangle$
$\langle e \rangle$	$\langle (-f)(ab)(df)cb \rangle, \langle (af)cbc \rangle$	$\langle e \rangle, \langle ea \rangle, \langle eab \rangle, \langle eac \rangle, \langle eacb \rangle, \langle eb \rangle, \langle ebc \rangle, \langle ec \rangle, \langle ecb \rangle, \langle ef \rangle, \langle efb \rangle, \langle efc \rangle, \langle efc b \rangle$
$\langle f \rangle$	$\langle (ab)(df)cb \rangle, \langle cbc \rangle$	$\langle f \rangle, \langle fb \rangle, \langle fbc \rangle, \langle fc \rangle, \langle fcb \rangle$

**Fig.20.** the projected database and sequential pattens

At the beginning of this step, we find sequential patterns having prefix about above-mentioned six items respectively. Take the item a as the example, the sequences which include this item would be collected, and we just consider the subsequence prefixed with the first occurrence of a.

Then, we scan a-projected database once in order to find all the length-2 sequential patterns. Recursively, we continue to find the non-empty postfix subsequences having prefix length-2 sequences. As same as the item a, we find sequential patterns having prefix item for b, c, d, e and f respectively.

### the Efficiency of the Algorithm

There are three aspects about the efficiency of PrefixSpan algorithm. Specifically, there isn't any candidate sequence generated by this algorithm. Based on this feature, PrefixSpan is faster than Apriori-like algorithm, and it searches a much smaller space than GSP algorithm, which would generate and test a substantial number of candidate sequences. Secondly, the projected database is smaller than the original one due to the algorithm just project the postfix subsequences of the frequent prefix into the projected database. Therefore, the projected databases keep shrinking.

The above-mentioned is the advantages of the PrefixSpan algorithm, now we talk about the drawback about it. Construct the projected databases is time-consuming job, because sometime the algorithm would build the projected database for all of the sequential patterns.

The pseudo-code of the FPClosed-growth algorithm is as shown as Fig.21.

### 3.5.2 the FPClose Algorithm

The FPClose algorithm is an algorithm in the FPGrowth algorithm family and uses another variant of the FP-tree structure called CFI-tree (closed frequent itemset-tree). This algorithm is designed for mining frequent closed itemsets based on the FPGrowth algorithm, and it may be one of the fastest closed itemset mining algorithm.

#### CFI-tree

In order to introduce the FPClose algorithm, we explain CFI-tree firstly. CFI-tree, closed frequent itemset tree, it depends on FP-tree and is denoted as CX. This CFI-tree construction always stores all of the closed frequent itemsets including itemset X and their counts which were found. A new frequent itemset Y including X which was found, we need to compare it with the CFIs in CX. And the concept of closed frequent item can be found in the second chapter. Each node of a CFI-tree have four attributions, they are item-name, count, node-link, and level, respectively.

Item-name and node-link have same meaning in FP-tree, and the term, level, is used to test the subtree as same as the purpose in MFI-trees. Due to we need to compare Y with a set Z in the tree, so that confirm Y and Z don't have the same count where Y belong to Z, then the count term is essential. The operation of inserting a CFI into a CFI-tree is similar to the operation of inserting a transaction into a FP-tree. However, there are different between

```

Procedure FPclose(T, C)
Input: T, an FP-tree
       C, the CFI-tree for T.base
Output: Updated C
Method:
1. if T only contains a single branch B
2.   generate all CFI's from B;
3. for each CFI X generated
4.   if not closed_checking(X, C)
5.     insert X into C;
6. else for each i in T.header do begin
7.   set Y = T.base ∪ {i};
8.   if not closed_checking(Y, C)
9.     if T.FP-array is defined
10.      let tail be the set of frequent items for i in T.FP-array
11.     else
12.      let tail be the set of frequent items in i's conditional pattern base;
13.     sort tail in decreasing order of items' counts;
14.     construct the FP-tree TY and possibly its FP-array AY;
15.     initialize Y's conditional CFI-tree CY;
16.     call FPclose(TY, CY);
17.     merge CY with C;
18. end

```

**Fig.21.** The FPClose Algorithm

these two tree construction. The count of a node would not increase instead of replacing by the maximal count up-to date in CFI-tree. [8]

### the Efficiency of the Algorithm

- the Runtime

In most cases, the number of CFIs is greater than the number of MFIs, although still far less than the number of FIs. Now, we cite the result about the runtime of mining CFIs on T100I20D200K from the <Fast Algorithms for Frequent Itemset Mining Using FP-Trees> as shown as Fig.21.

In the graph, we can see the result about the runtime of all algorithms on the synthetic data set. The FPClose algorithm is the fastest algorithm than others in case of the minimum support value, however, it becomes slower than Apriori and CHARM in high minimum support(the minimum support value which is equal to 6 is the threshold). The reason why FPClose algorithm run more slowly than those two algorithm is that construct a wide FP-tree is a time-consuming job, especially, the FP-tree yields only small number of CFIs. Obviously, we can find that the CLOSET+ algorithm is the slowest in any condition.

On the other hand, CHARM and Apriori algorithm store and generate the candidate FIs and CFIs by small sized data structures. Thus, the FPClose algorithm becomes the fastest algorithm in the low minimum support condition, due to this algorithm just run based on the first stage pays off. Then in case of the low minimum support for Apriori and others must deal with a lot of candidate FIs. [8]

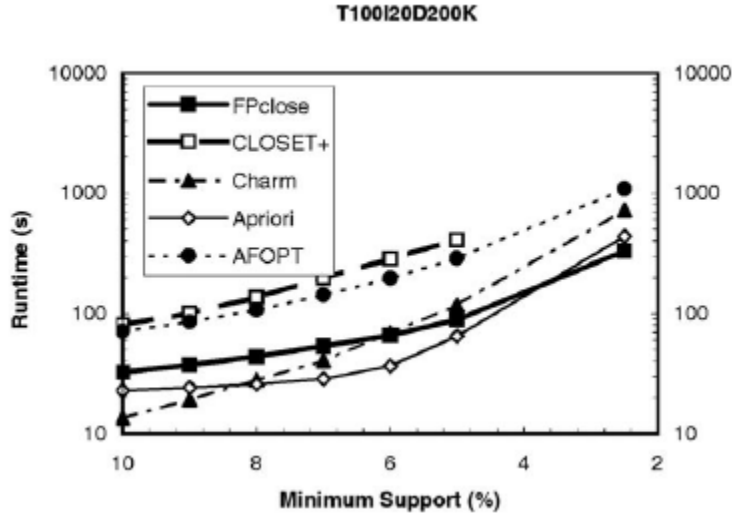


Fig.21. Runtime of mining CFIs on T100I20D200K

- Memory Consumption

The FP-growth method which is the basic of FPclose algorithm, it uses the amount of memory. And construct the FP-tree would consume much memory when running on a synthetic data set.

### 3.6 Conclusion

In this chapter, we proposed the whole solution to generate the frequent sequences for each course in our project. In the solution, we generated the original sequences, and then filter, cleanse and transform them to adapt any requirement of data mining, all of the above-mentioned operation are based on Java platform. Then, we carried out the frequent pattern mining use an open-source data mining library based on Java platform. Finally, we calculate the alignment distance between the test sequence and frequent sequences for each course.

# Chapter 4

## Experiments

In this chapter, we will test our solutions based on Eclipse platform. At first, we will pre-process the data set, cleanse and transform it using Python language. Then, run the data mining code to obtain the frequent sequences for each course using Java language. Finally, compute the similarity of one user sequence to each frequent sequence of the success and failed user, then keep the maximum similarity. From the two scores, cluster the user sequence in the failed user or succeed user.

### 4.1 Dataset

In this project, we carry out data mining from four different courses. They are JAVA, Node.js, XML and Audace Entrepreneure, respectively. The following table presents the basic information from the database.

Table Name	the Number of Records		
course	367	the Name of Type	Quantity
		course	4
		part	30
		chapter	99
		section	234
premium	3992		
quizz	169133		
session	489385		
subscription	33636	the Name of Course	Quantity
		JAVA	26163
		Node.js	3309
		XML	1565
		Audace entrepreneure	2599
user	29650		

In the course table, we had integrated Chapter and chapter into chapter, and Part, partie and subchapter into part. Then we get this table contains 367 records for four types (4 types course, part, chapter, section with respectively 4, 30, 99, 234). And There are 33636 records in the subscription talbe for four courses (JAVA, Node.js, XML and Audace entrepreneure with respectively 26163, 3309, 1565, 2599). Moreover, The number of records in



the premium, quizz, session and user table is 3992, 169133, 489385 and 29650 respectively.

We can see that there are more than 600,000 records for this large set of dataset. Among this dataset, we need to find out some features to build the user profile.

## 4.2 Pre-processing

Due to the original sequences which generated from the database is not suitable for data mining, we need to carry out data pre-processing for them. The format of these sequence is a multi-dimensional list (the format was mentioned in last chapter), so it's inconvenient for us to mine each list, set, tuple and element. Therefore, we need to filter sequences. We prune these multi-dimensional list into a low-dimensional list, a set of list, list, tuple and element, then assemble them into different specific format according to the requirement for data mining. Actually, this operation step is a dull and time-consuming work, however, it's also the essential step for the follow-up work.

At first, we remove the courseid in the original multi-dimensional list, this operation reduces one dimensionality of the list. There only are four courses and it's easy to distinguish the course ID, these is the reason why we can remove the courseid attribution. Then, we would get a set of multi-dimensional list which contains the reading session for a user, it means we can analysis reading sessions for different user separately.

In this moment, we found there are some unreasonable factors in each user's reading session list. The list exist the following problems:

- The course type of partid should be removed. Because the course type of partid would be record into the database when a user start to learn this course each time, it's unuseful for analysis user's reading session.
- Due to a set of consecutive reading actions from a learner that should be considered continuous (apart from small interruptions, e.g. for reading email). [18] Thus, the same idPart may appears more than once with the same sessionid in different tuple, this kind of phenomenon exists in list actually. Therefore, we need to remove the reduplicative idPart, assemble the idSession and accumulate the duration value in this case.
- After the previous operation, we need to sort those lists. Finally, we generate a new idPart list group by session id.

## 4.3 Data Mining

After these pre-processing operation, we can extrace many user's feature for the filtered sequences now. The data mining results are presented in the following

### 4.3.1 Subscription Data

From the subscription information, we can find the number of subscriptions with a non empty score is 194 ( courses Java, Node.js, XML, AudaceEntreprendre with respectively 59, 76, 18, 41 non empty scores ) and the distribution of the whole subscriptions is shown as the following table( the maximun of the subscription is 100 ).

Quantity	Value Ranges
3(41, 22, 41)	[0, 50[
2	[50, 60[
12	[60, 70[
35	[70, 80[
47	[80, 90[
85	[90, 100[
10	100

The average score of the whole reasonable subscriptions in the data set is equal to 85.96. And we also can mine the average score for each courses, the result are shown as following table:

CourseName	Score	Quantity
Java	84.60	59
Node.js	92.05	76
XML	87.33	18
AudaceEntreprendre	76.02	41

### 4.3.2 Reading Session Data

For the reading session information, we can mine the number of users with a reading session is 12989. The number of total users from four course is bigger than the number of users due to many users register more than one course. For more details about these value, we can see in the following table:

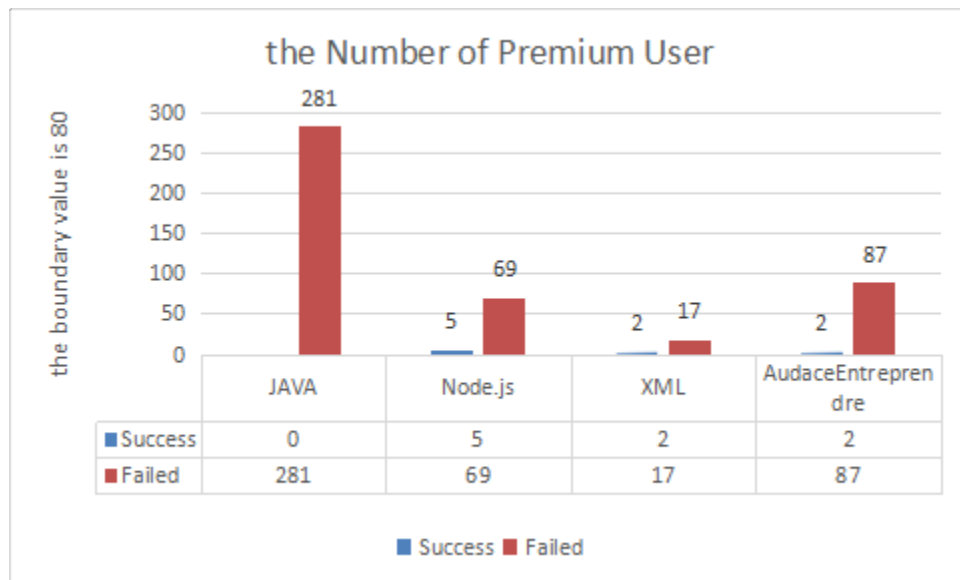
CourseName	Quantity
JAVA	10026
Node.js	1844
XML	1874
AudaceEntreprendre	1824

The number of sessions for the courses Java, Node.js, XML and Audace Entreprendre is 408805, 25350, 36276 and 18954 respectively.

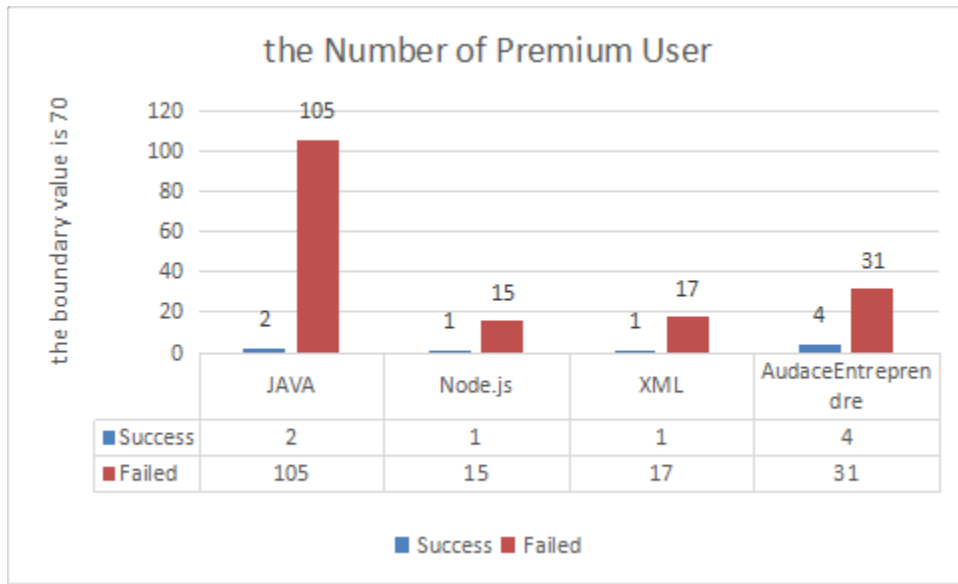
### 4.3.3 Premium Data

For the premium users information, we found out that there are 3573 distinct users registered the premium service, 3992 registrations of premiums, 419 registrations of users with more than one registration operation and 66 users with more than one registration.

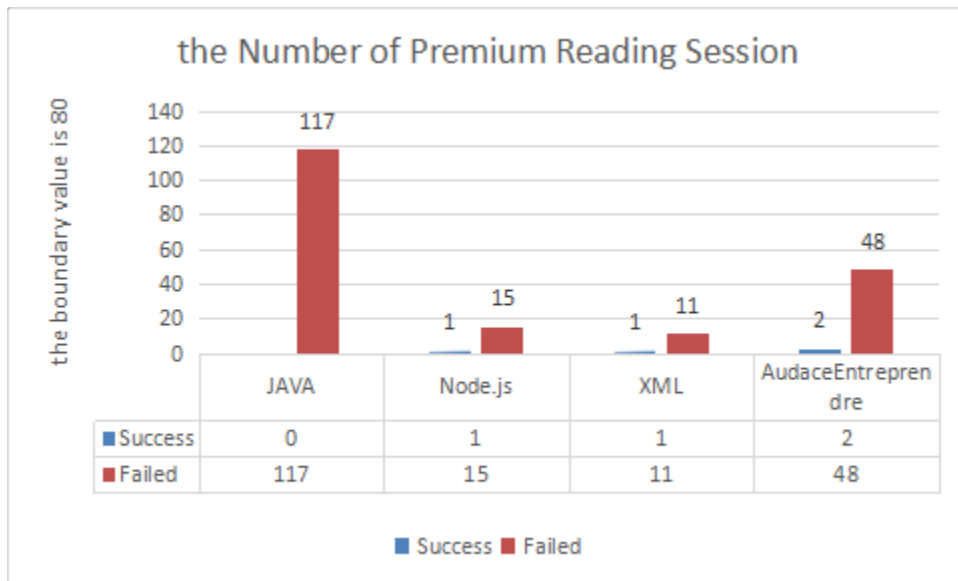
There are two ways of distinguishing the success and failed users. Firstly, we set the critical value into 80, and we can see the detail number for each course and each kind:



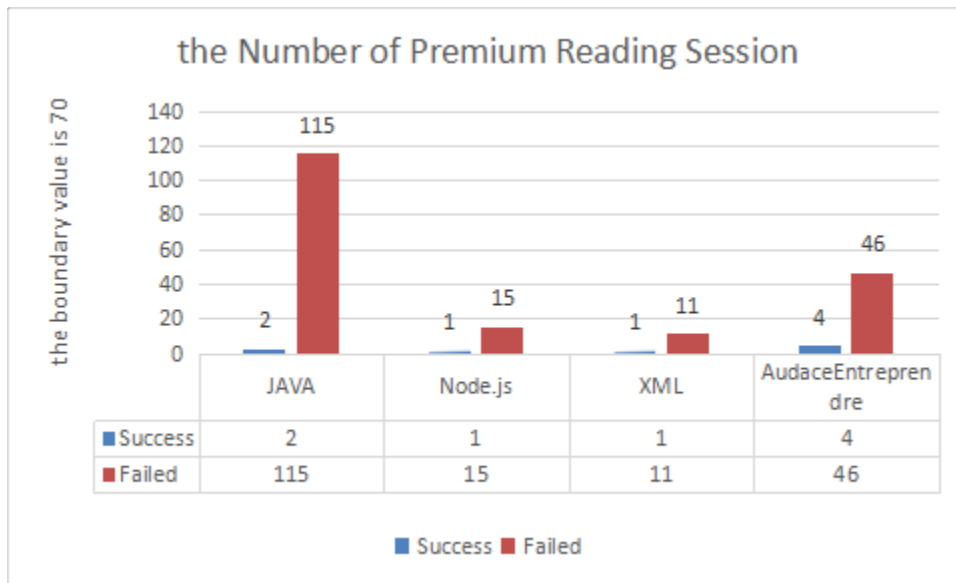
Secondly, we set the critical value into 70, it's also the standard critical value in OpenClassrooms website.



For the premium user in case of reading sessions, it's also an important factor for frequent mining. As same as the distinguishing rule for the number of the premium users, we divided these values into 2 groups:

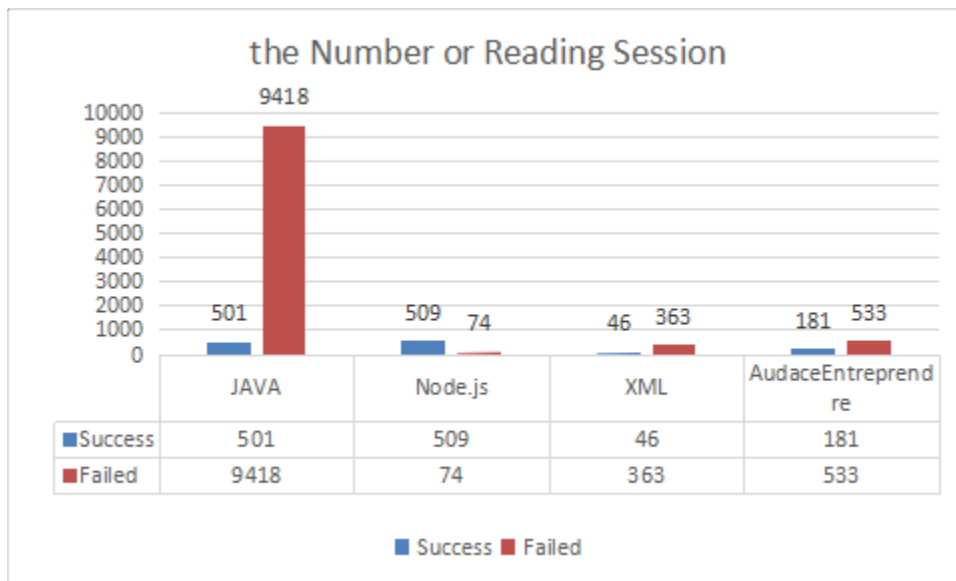


For a boundary  $score \geq 70$ .



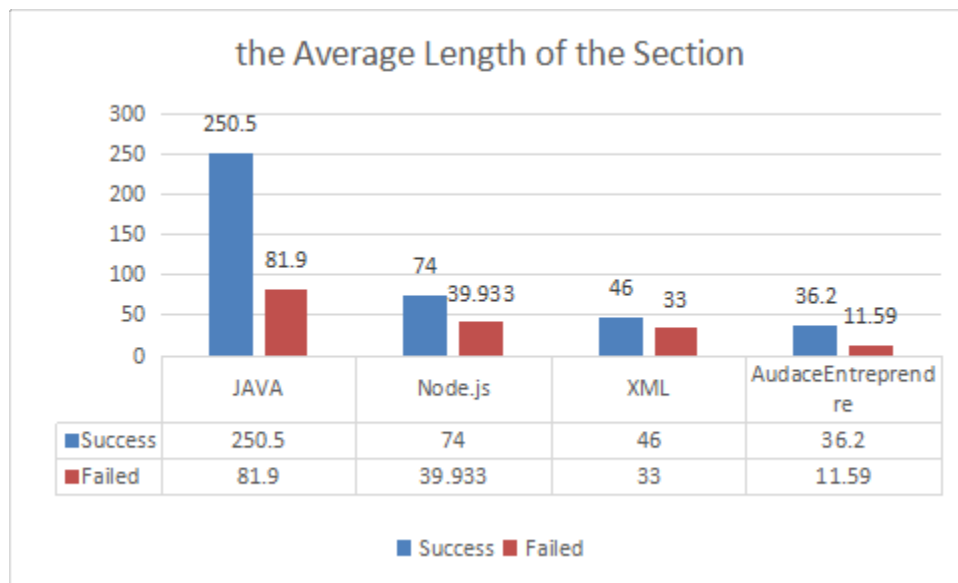
#### 4.4 Reading Section

A reading session has 4 type included course, part, chapter and section. In this project, we should remove the course session, because the course session would appear each time when the user start to learn the course. After the previous operation, we get the reading section. Now we set the boundary into 70 and get the number of reading section for each course as following:



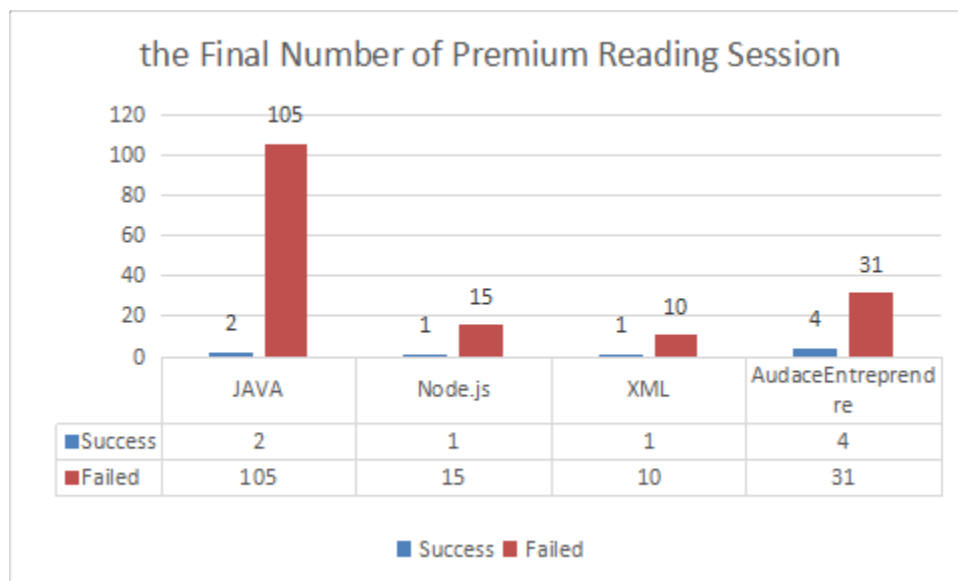
#### 4.5 Course Data

Moreover, we can get the average length of the sections for each course. For more detail, it is shown as following:



## 4.6 Generate Frequent Sequences

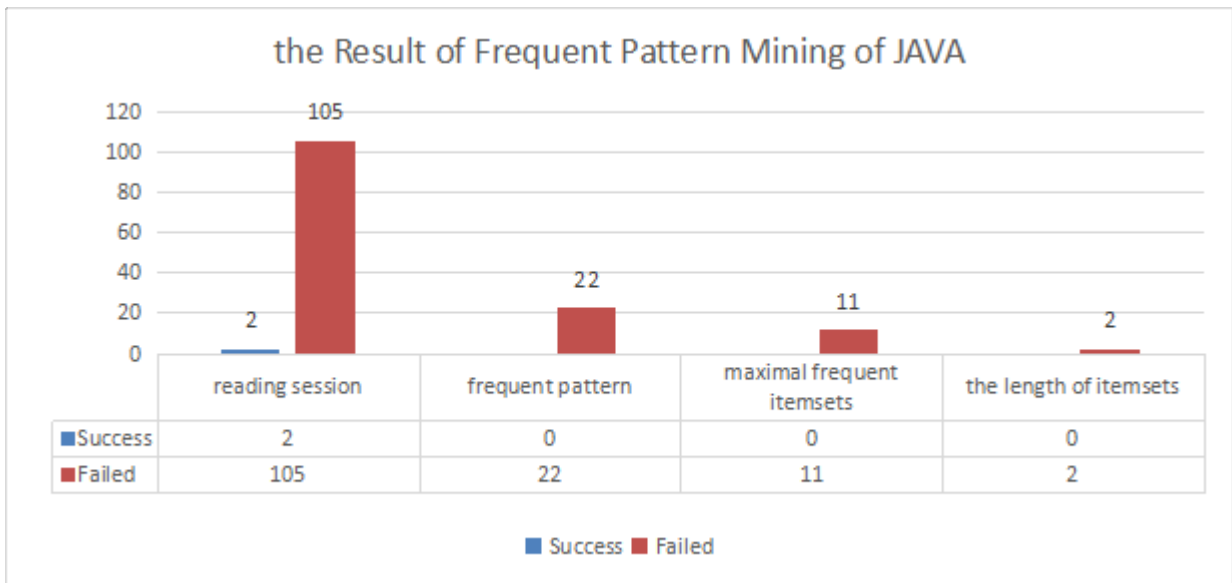
Based on the previous section, we had transferred the original format into the specific format which is suitable for the algorithm, the final number of premium reading session is as shown as the following figure. And we know how many reading session sequences for each kinds of user from each course. Then, we use the PrefixSpan algorithm to obtain their frequent itemsets, and set the minimum support value to 0.5. Due to keep more complete information of the reading session, we would just extract the maximal frequent itemsets from the generated frequent patterns.



### 4.6.1 the Result of FPM

- JAVA

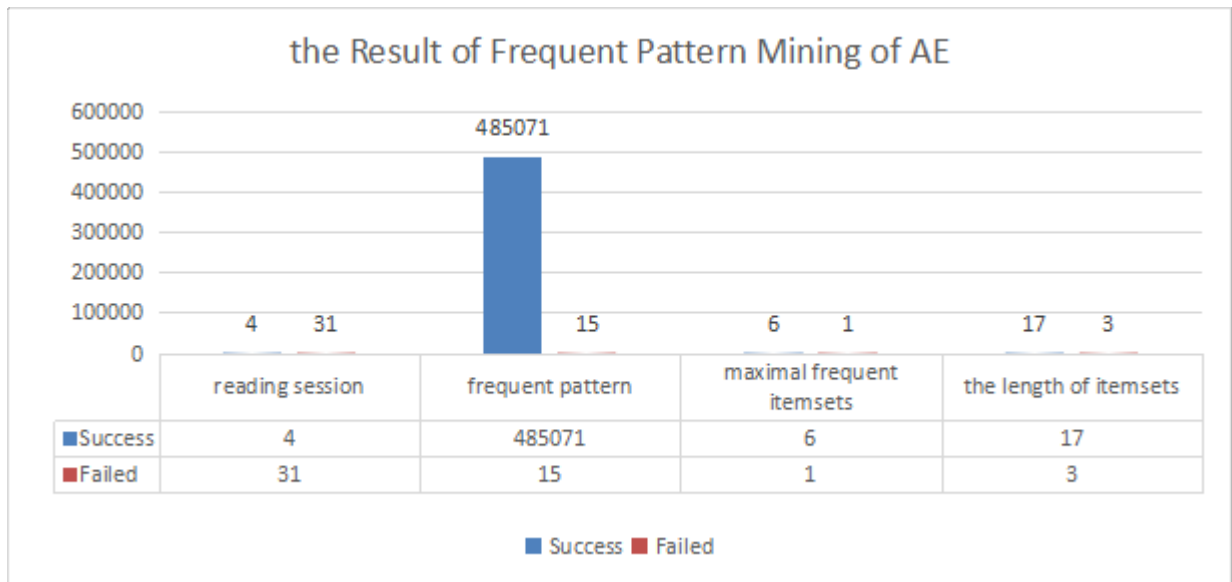
In the JAVA course, there is not any pattern found for success users, and we found 22 patterns for failed users. Then we can extract 11 maximal frequent itemsets to analysis and their length is 2. Moreover, the support of the pattern as a number of sequences is from 53 to 61, and the frequent itemset with the maximal support is [20304,20304]. We present the detail data in the following figure.



- Audace Entrepreneure

In the Audace Entrepreneure course, there are 485071 patterns found for success users, and 15 patterns for failed users.

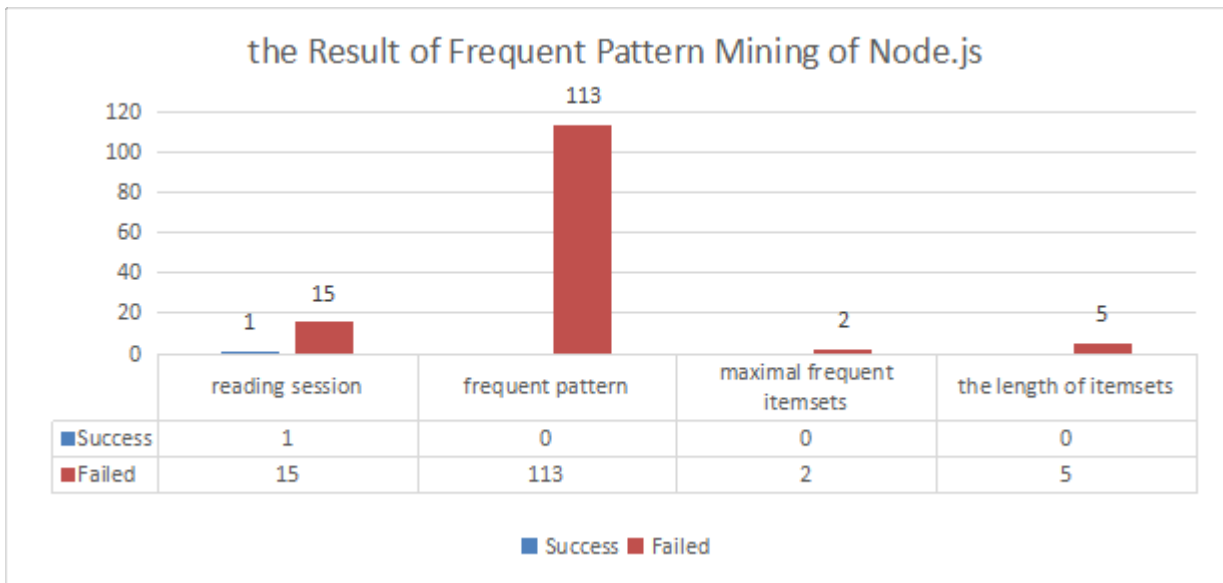
In case of success users, we can extract 6 maximal frequent itemsets. All them, the support value is 2 and the length is 17. On the other side, we can obtain 1 maximal frequent itemsets and the support value and the length is 19 and 3 respectively. We present the detail data in the following figure.



- Node.js

Due to there just is one reading session for success users in this course, it's not possible that it exists any frequent pattern. Moreover, there isn't maximal frequent itemsets and the length value about it.

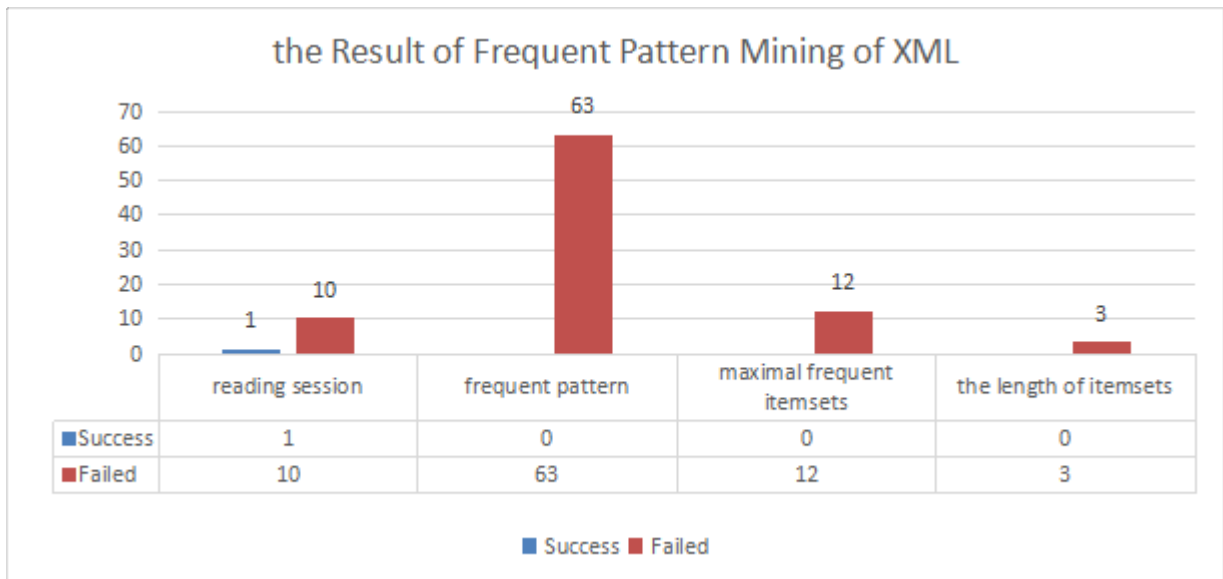
On the other hand, there are 15 reading session for failed user. Thus we can find 113 frequent patterns, obtain 2 maximal itemsets and their support value is 8 and 9 respectively. The length of these MFI is 5. We present the detail data in the following figure.



- XML

As same as the Node.js course, the result of FPM for success user doesn't contain the frequent pattern, maximal frequent itemsets and the length value.

On the other hand, there are 10 reading session for failed user. Thus we can find 65 frequent patterns, obtain 12 maximal itemsets with same support value is 5. The length of these MFI is 3. We present the detail data in the following figure.



## 4.7 Compute Sequence and Alignment Distance

In this section, we would compute some parameter between a user's reading session and the frequent patterns. At first, we would create the above-mentioned two sequences to be aligned and encode them.

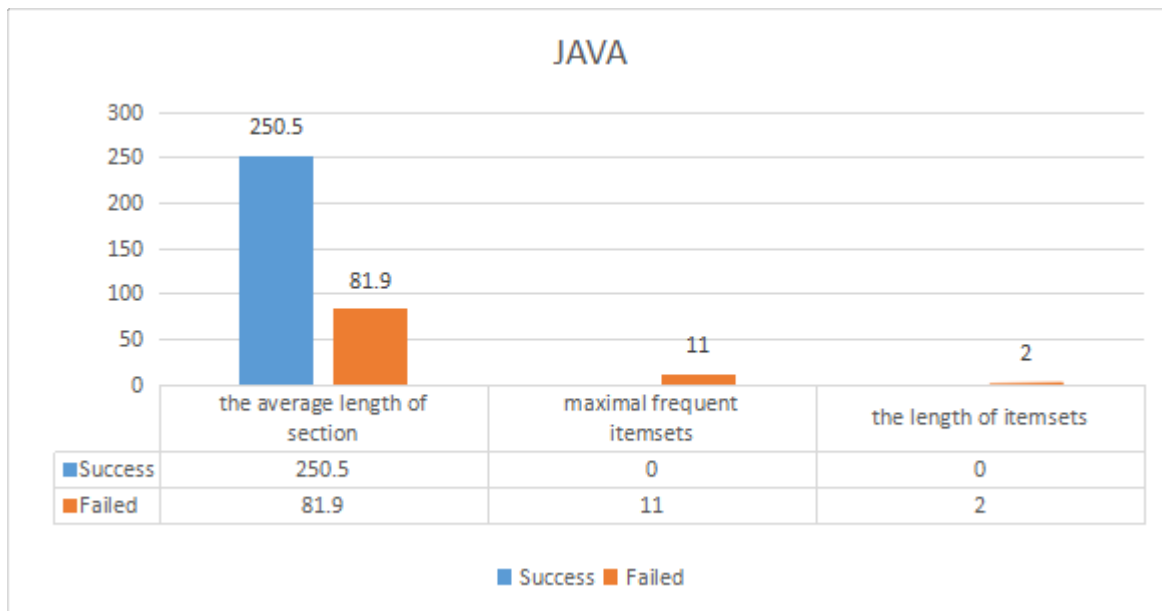
Secondly, due to there are two kinds of alignment algorithm, they are global and local alignment algorithm, respectively. Thus, we would obtain the optimal global and local alignments.

Thirdly, for the scoring system in alignment algorithm, it defines how to score how each individual pair of letters match up. We set the matching score, mismatching score and gap score into 10, -1 and -1 in our global and local scoring system, respectively. Thus, we obtain the global alignment score, local alignment score and their percent identity.

Finally, we obtain the sequence distance between these two sequence. Now we present the experiments result for four courses detailly.

- JAVA

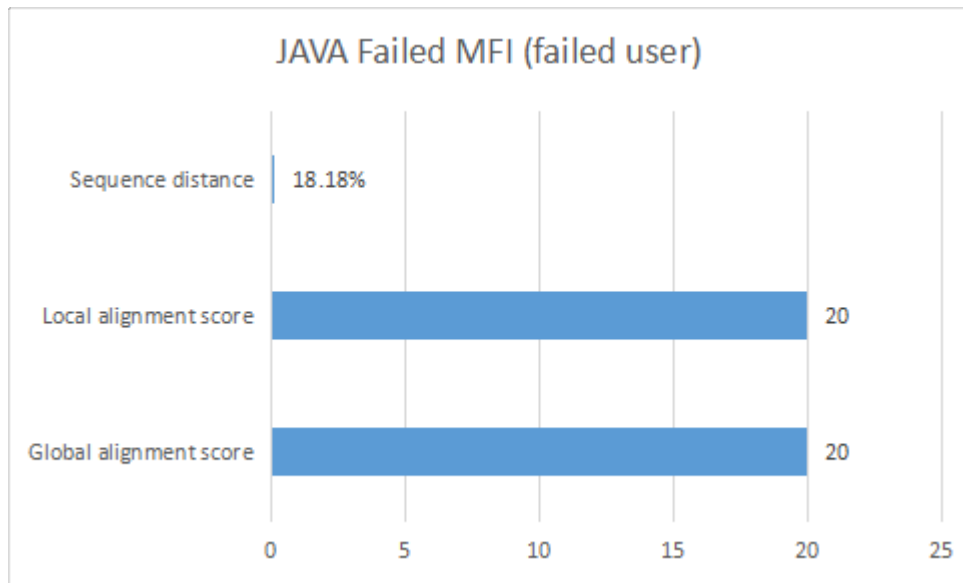
Now we assemble and extrace the useful information provided by the previous several section, and present it in the following graph:



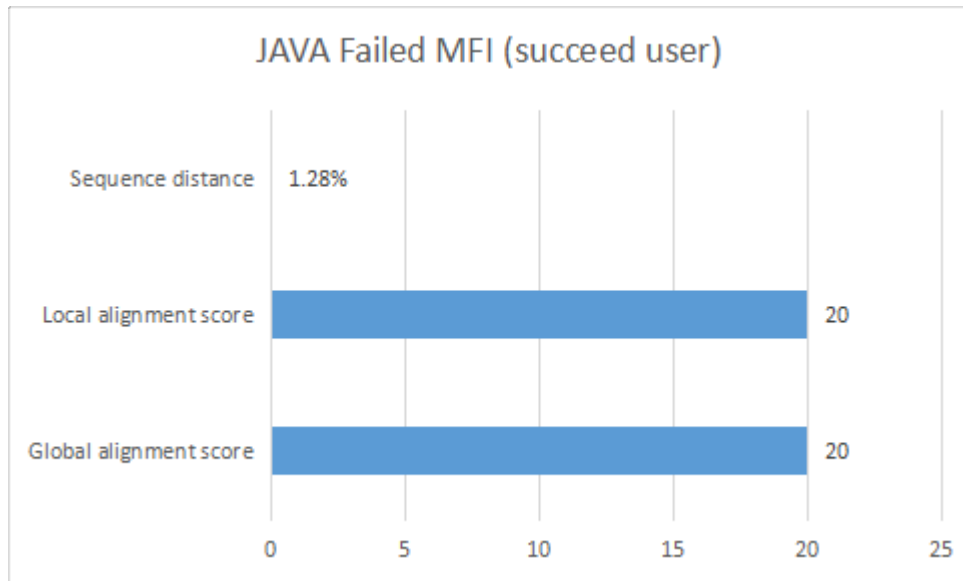
Firstly, we select the user who isn't a premium user, register the JAVA course, get the mark, and has the reading session( Actually this user's mark is 0). These selections are important due to it's not included in frequent pattern mining procedure, then this sequence can test the solution better. The sequence is [20304, 20564, 20615, 20757, 20841, 20903, 20998, 21114, 21282, 21424, 21530, 21530, 21583, 21973, 22107, 22162, 22278, 22404, 22756, 22839]. It's obvious that its length is 20, and this value is closed to the case of failed, it means this user is more likely to be a failed user.

Secondly, we carry out the experiment about sequence and alignment distance. We compare this sequence with the frequent patterns for the JAVA failed users. And just keep the results with maximal similarity. We can obtain the optimal global alignments and optimal local alignments when all of the their score, percent identity and sequence distance reach the maximal value respectively. Then, we found out that the optimal global alignments is as same as optimal local alignments. There are 6 optimal alignments, [20304, 20564], [20654, 20615], [20564, 20615], [20615, 20757], [20757, 20841] and [20841, 20903] respectively. And the experiment result is shown as the following graph which topic is JAVA Failed MFI(failed user).





Due to there isn't frequent sequence for success users, we select another user to do the same procedure with the same selection condition( this user's mark is 95). Its length of section is 311 and this value is closed to the case of success user. Compare this sequence with frequent sequences for failed users. Then, we can see the experiment result is shown as the following graph.

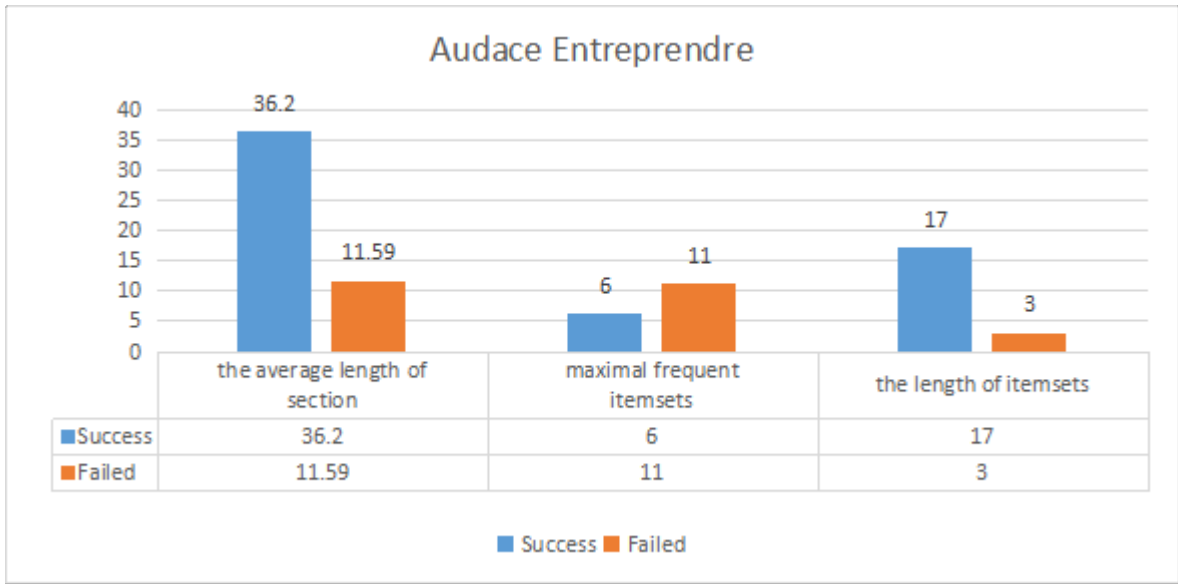


In this case, because the frequent sequences about success users and failed users don't appear at the same time, we compare the sequences from the success and failed user with the frequent sequences generated from failed user. We found out the sequence distance values are the most obvious value, one of them is equal to 0.182 and the other is equal to 0.01. It's obvious that a failed user get the higher value than a success user, when compare them with frequent sequences for failed users. Thus, we can cluster a user sequence in the different user group based on the higher sequence value and the closer length of the reading section.

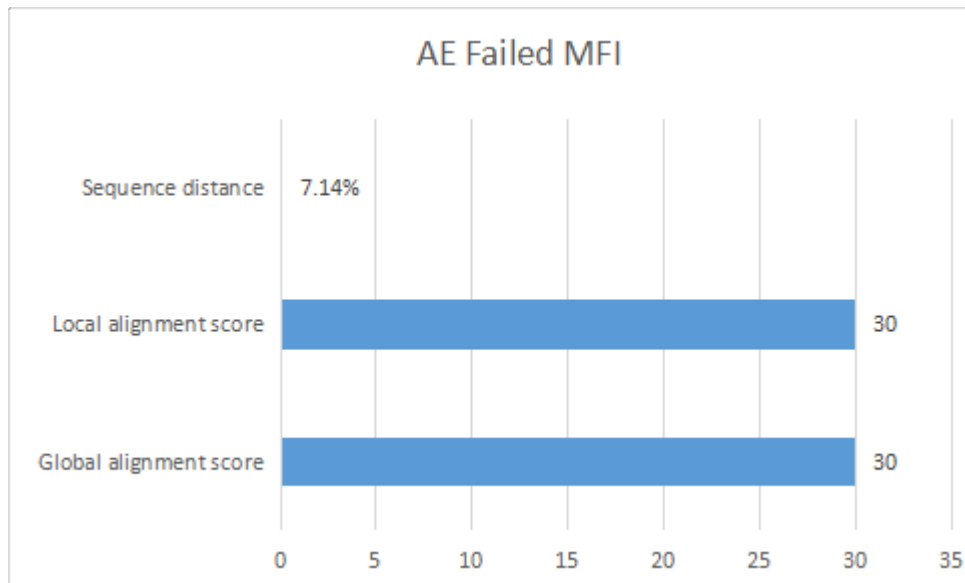
- Audace Entreprendre

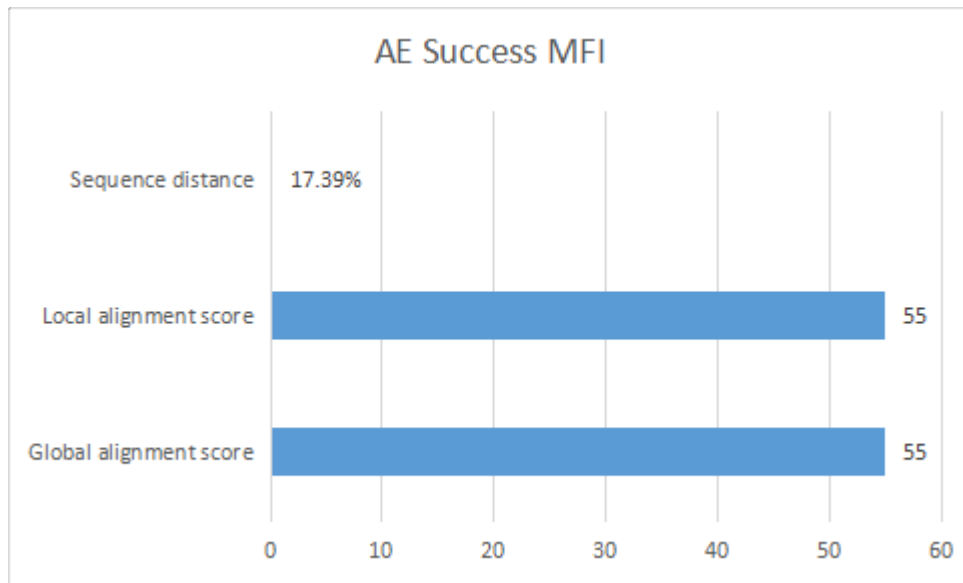
Firstly, we select the user (this user's mark is 83 as a success user) who isn't a premium user, register the Audace Entreprendre course, get the mark, and it's obvious that its length is 81, and this value is closed to

the case of success, it means this user is more likely to be a succeed user.



Secondly, we carry out the experiment about sequence and alignment distance. We compare this sequence with the frequent patterns for the Audace Entrepreneure failed users, then compare it with the frequent patterns for the success users. We can see the experiment result as shown as the following two graphs.

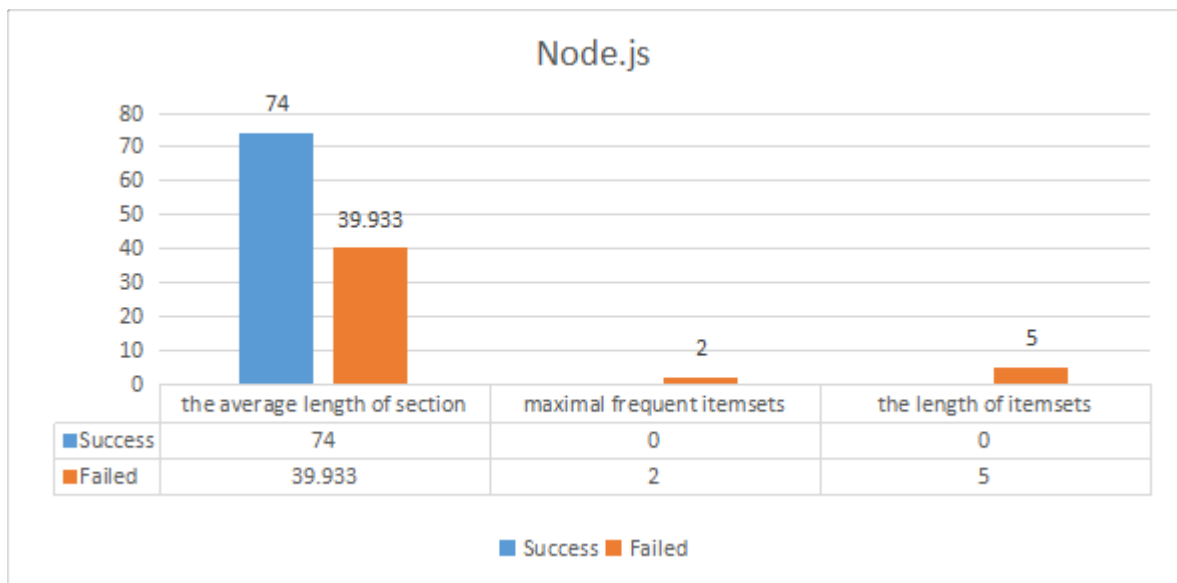




Compare these two above-mentioned graph, we found out all of the parameters (which are generated by comparing the frequent patterns for success users) are higher than the other one. Put it in detail, the sequence distance for success frequent sequences is 10 percent higher than failed frequent sequences, and both of the local and global alignment score is almost twice as the result generated from the failed frequent sequences. Finally, we can cluster this user sequence in the success users (actually he is a success user).

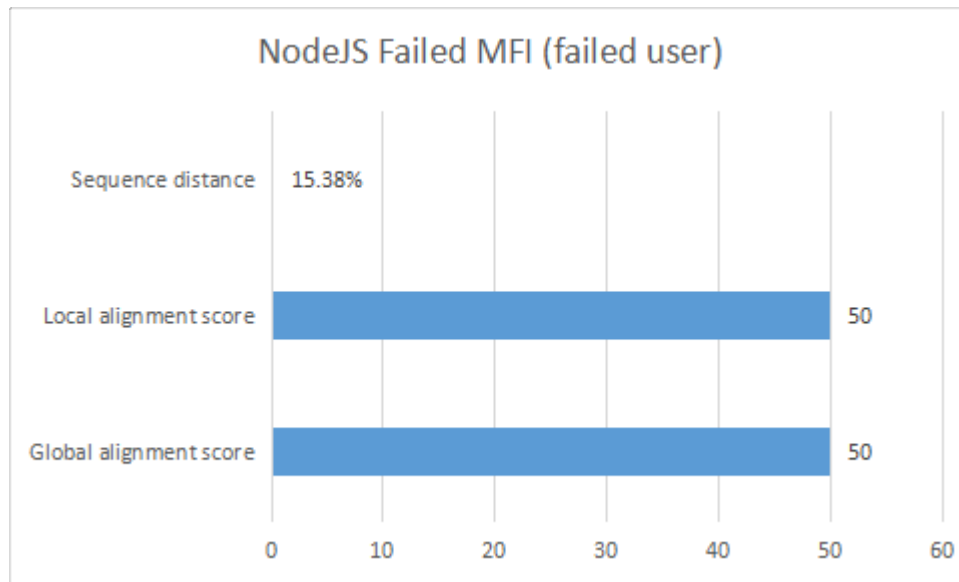
- Node.js

We can see the fundamental information in the Node.js course. This course and the JAVA course are in a similar situation. We want to mine the data based on only one kind of frequent sequences as many as possible, therefore, we need choose two kinds of users to compare with these frequent sequences of failed users.

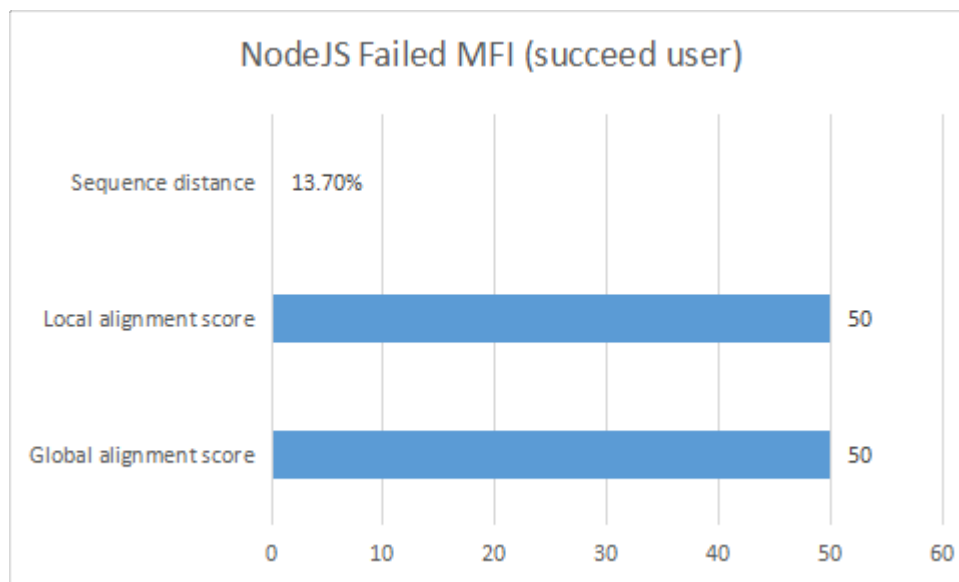


Firstly, we select the user whose actual mark is 0, and its length is 60, and this value is closed to the case of failed. It means this user is more likely to be a failed user, but that's not actually.

we carry out the experiment about sequence and alignment distance. We compare this sequence with the frequent patterns for the NodeJS failed users. And just keep the results with maximal similarity. And the experiment result is shown as the following graph.



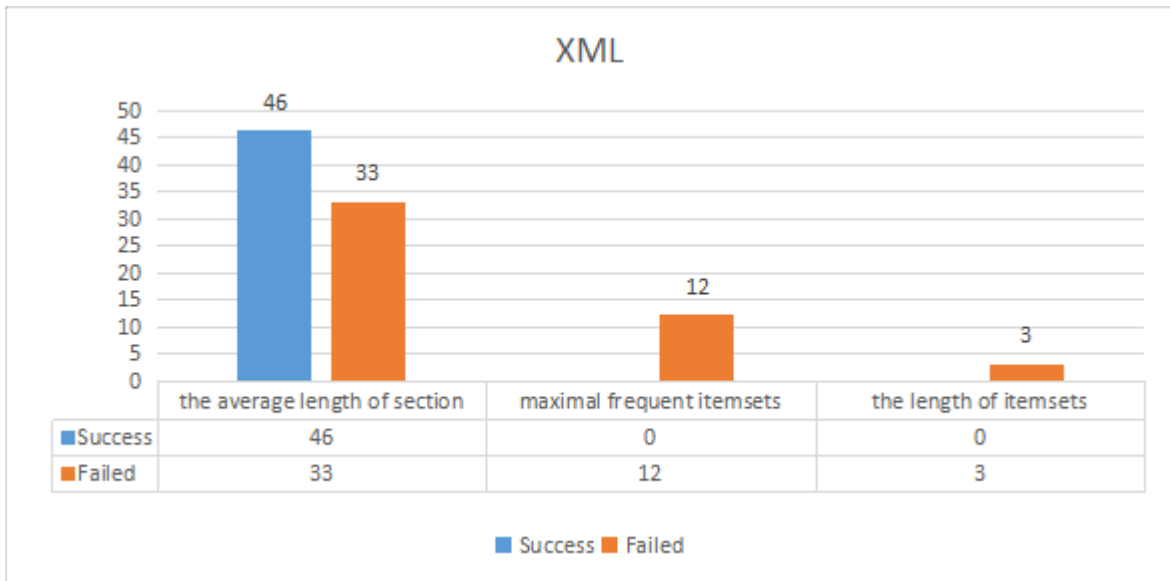
Secondly, due to there isn't frequent sequence for success users, we select another user to do the same procedure with the same selection condition( this user's mark is 100). Its length of section is 68 and this value is closed to the case of success user (the value is 74). Compare this sequence with frequent sequences for failed users. Then, we can see the experiment result is shown as the following graph.



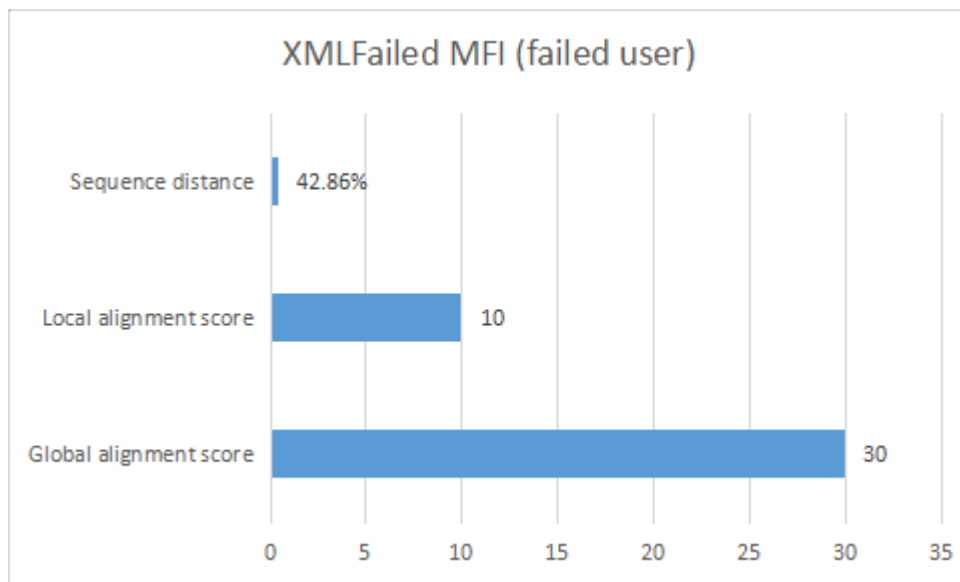
As same as the JAVA course, we compare the sequences from the success and failed user with the frequent sequences generated from failed user. We found out the sequence distance values are the most obvious value, one of them is equal to 0.1538 and the other is equal to 0.137. These two values are so similarity, it's easy to find that the number of the frequent sequences and the test sample aren't enough, then they causes this problem. However, the failed user still obtain the slight advantage about the experiment result than the succeed user.

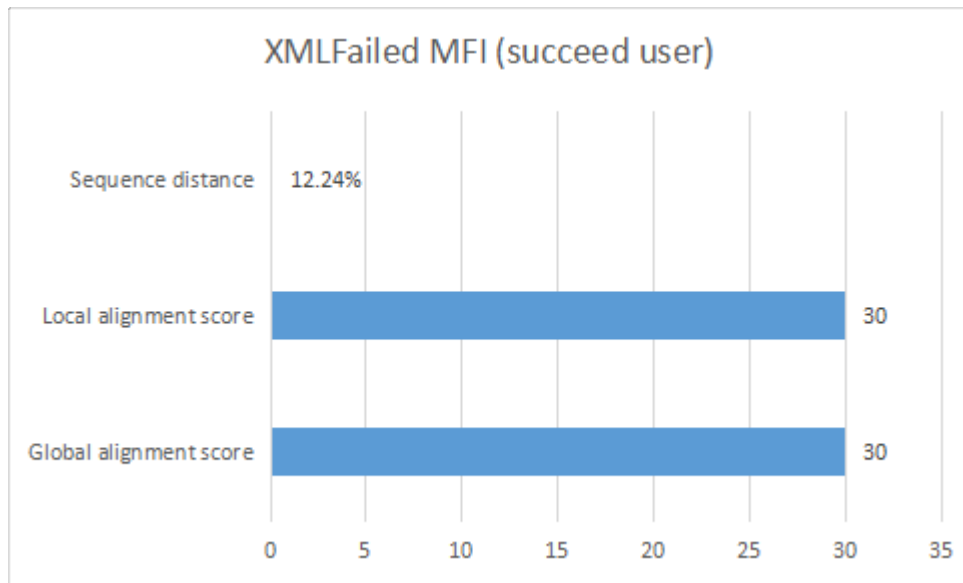
- XML

This course and the JAVA course are in a similar situation. We can see the fundamental information about the XML course in the following graph.



We still choose two kinds of user to compare. The first one is a succeed user whose mark is 87 and length of reading session is 46. And the second one is an actual failed user whose mark is 0 and length of reading session is 11. Now we present the experiment result shown as the following two graphs.





We found out that there is a tremendous differences between these two sequence distance value. The sequence distance is generated from failed user is 30 percent higher than it's generated from success user. What is more, the value is equal to 42.86 percent, it means the frequent itemset and this sequence of failed user are all very similar. Thus, we can cluster this user to the failed user group without question.

## 4.8 Conclusion

In this chapter, we test the whole solution proposed in Chapter 3 on four course. Use the PrefixSpan algorithm to generate the frequent patterns instead of the FPClose algorithm, because construct the CFP-tree is a time-consuming work. The runing time would be increase with the the support increases, especially, run it in this large set of data set. Hence, we use the previous algorithm to obtain the maximal frequent itemset, and use them to compare with the test sequence, then calculate their sequence distance and make them aligned. Through the experiment, we found out the sequence distance is the important parameter to cluster a user into different group.

At first, we want to compute the similarity of one user sequence to each frequent sequence of the success user, and do the same operation to failed user. However, only the Audace Entreprendre course satisfies this condition due to it has two kinds of frequent sequence and the data sample is big enough. Moreover, for the other courses, we don't have enough reading session to generate frequent itemset. The reason is that the number of premium success user of JAVA, Node.js and XML is too small, 2, 1 and 1 respectively. Then we use another method to finish the goal, we choose two kinds of user to compare with the frequent sequences instead of the previous method. Due to the above-mentioned problem, it's not ideal in the Node.js course. However, in the Java, Audace Entreprendre and XML course, we all found out the obvious different from two different experimental sequences and cluster the test user into the specific group without bias.

# Bibliography

- [1] Charu C Aggarwal and Jiawei Han. *Frequent pattern mining*. Springer, 2014.
- [2] Stephen F Altschul and Bruce W Erickson. Optimal sequence alignment using affine gap costs. *Bulletin of mathematical biology*, 48(5-6):603–616, 1986.
- [3] Abhang Swati Ashok and S JoreSandeep. The apriori algorithm: Data mining approaches is to find frequent item sets from a transaction dataset. *International Journal of Innovative Research in Science, Engineering and Technology*, 3, 2014.
- [4] Teresa K Attwood and David J Parry-Smith. *Introduction to bioinformatics*. Prentice Hall, 2003.
- [5] Christian Borgelt. An implementation of the fp-growth algorithm. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, pages 1–5. ACM, 2005.
- [6] Daniel G Brown. A survey of sequence alignment. In *Computational Genomics: Current Methods*, 2006.
- [7] Philippe Fournier-Viger. An open-source data mining library, 2016.
- [8] Gösta Grahne and Jianfei Zhu. Fast algorithms for frequent itemset mining using fp-trees. *IEEE transactions on knowledge and data engineering*, 17(10):1347–1362, 2005.
- [9] Cornelia Györödi, Robert Györödi, and Stefan Holban. A comparative study of association rules mining algorithms. In *Hungarian Joint Symposium on Applied Computational Intelligence, Oradea*, 2004.
- [10] Jiawei Han. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth.
- [11] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM Sigmod Record*, volume 29, pages 1–12. ACM, 2000.
- [12] Leslie HUIN, Yann BERGHEAUD, Pierre André CARON, Alexandra CODINA, and Eric DISSON. Measuring completion and dropout in moocs: A learner-centered model. *Research Track*, page 55, 2016.
- [13] David J Lipman, Stephen F Altschul, and John D Kececioglu. A tool for multiple sequence alignment. *Proceedings of the National Academy of Sciences*, 86(12):4412–4415, 1989.
- [14] MS Mythili and AR Mohamed Shanavas. Performance evaluation of apriori and fp-growth algorithms. *International Journal of Computer Applications*, 79(10), 2013.
- [15] Openclassrooms. about-us, 2016.
- [16] Valery O Polyanovsky, Mikhail A Roytberg, and Vladimir G Tumanyan. Comparative analysis of the quality of a global algorithm and a local algorithm for alignment of two sequences. *Algorithms for molecular biology*, 6(1):1, 2011.
- [17] Sanjeev Rao and Priyanka Gupta. Implementing improved algorithm over apriori data mining association rule algorithm 1. 2012.
- [18] Madjid Sadallah, Benoît Encelle, Azze-Eddine Maredj, and Yannick Prié. Towards reading session-based indicators in educational reading analytics. In *Design for Teaching and Learning in a Networked World*, pages 297–310. Springer, 2015.
- [19] Wikipedia. Apriori algorithm.

[20] Wikipedia. Datamining.

[21] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.