The Data Mining Blog
A blog by Philippe Fournier-Viger about data mining, data science, big data...

An Introduction to Sequential Rule Mining

Posted on 2015-08-18 by Philippe Fournier-Viger

In this blog post, I will discuss an interesting topic in data mining, which is the topic of **sequential rule mining.** It consists of discovering rules in **sequences**. This data mining task has many applications for example for analyzing the behavior of customers in supermarkets or users on a website.

Introduction

Before, discussing this topic, let me talk a little bit about the context. There has been a lot of work in the field of **data mining** about **pattern mining** . The goal of pattern mining is to discover useful, novel and/or unexpected patterns in databases. In this blog post, we will be interested by a specific type of database called **sequences databases**. A **sequence database** contains some sequences. For example, consider the following database:

ID	Sequences
seq1	⟨{*a*, *b*},{*c*},{*f*},{*g*},{*e*}⟩
seq2	⟨{*a*, *d*},{*c*},{*b*},{*a*, *b*, *e*, *f*}⟩
seq3	⟨{*a*},{*b*},{*f*},{*e*}⟩
seq4	⟨{*b*},{*f*, *g*, *h*}⟩

A sequence database containing four sequences

This database contains four sequences named seq1, seq2, seq3 and seq4. For our example, consider that the symbols “a”, “b”, “c”, d”, “e”, “f”, “g” and “h” respectively represents some **items** sold in a supermarket. For example, “a” could represent an “apple”, “b” could be some “bread”, etc.

Now, a **sequence** is an ordered list of sets of items. For our example, we will assume that each sequence represents what a customer has bought in our supermarket over time. For example, consider the second sequence “seq2”. This sequence indicates that the second customer bought items “a” and “d” together, than bought item “c”, then bought “b”, and then bought “a”, “b”, “e” and “f” together.

Sequences are a very common type of data structures that can be found in many domains such as bioinformatics (DNA sequence), sequences of clicks on websites, the behavior of learners in e-learning, sequences of what customers buy in retail stores, sentences in a text, etc.

Discovering sequential patterns in sequences

An important data mining problem is to design algorithm for discovering hidden patterns in sequences. There have been a lot of research on this topic in the field of data mining and various algorithms have been proposed.

In the following, Iwill discuss two types of patterns that can be found. Iwill first discuss sequential patterns. Then, I will explain some of their limitations and then discuss sequential rules.

A **sequential pattern** is a subsequence that appear in several sequences of a database. For example, the sequential pattern <{a}{c}{e}> appears in the two first sequences of our database. This pattern is quite interesting. It indicates that customers who bought {a}, often bought {c} after, followed by buying {e}. Such a pattern is said to have a **support** of two sequences because it appears in two sequences from the database. Several algorithms have been proposed for finding all **sequential patterns** in a database such as **CM-SPADE**, **PrefixSpan** and **GSP**. These algorithms takes as input a *sequence database* and a *minimum support threshold (minsup)*. Then, they will output all sequential patterns having a support no less than *minsup*. Those patterns are said to be the *frequent sequential patterns*.

For example, for the above example, if we run **CM-SPADE** with ***minsup = 3***, we will find the following frequent sequential patterns:

```
<{a}> with a support of 3 sequences
<{a},{e}> with a support of 3 sequences
<{a},{f}> with a support of 3 sequences
<{b},{e}> with a support of 3 sequences
<{b},{f}> with a support of 4 sequences
```

Sequential patterns can be quite interesting. In the example, we can learn that buying item “b” is followed by buying item “e” in 3 sequences. However, **sequential patterns can be misleading**. An important limitation of sequential patterns is that there is no assessment of the probability that a pattern will be followed. Let me explain this in more details. For example, if we consider again the pattern <{b},{e}>. This pattern is said to appear in 3 sequences. It may thus seems likely that if someone buy “b”, he will also buy “e” after. But how likely? We can observe that item “b” appears in four sequences. Thus, the probability that “e” appears after “b” is actually 3 / 4 = 75 % (i.e. P(e|b)= 75%). But sequential patterns only indicate how often the pattern appears. They do not provide any indication about this probability.

Discovering sequential rules in sequences

This now lead us to the main topic of this post which is **sequential rule mining**. Sequential rule mining has been proposed as an alternative to sequential pattern mining to take into account the probability that a pattern will be followed. I will provide a few definitions and then we will look at a full example.

A **sequential rule** is a rule of the form **X -> Y** where X and Y are sets of items (itemsets). A rule X ->Y is interpreted as if items in X occurs (in any order), then it will be followed by the items in Y (in any order). For example, consider the rule {a} -> {e,f}. It means that if a customer buy item “a”, then the customer will later buy the items “e” and “f”. But the order among items in {e,f} is not important. This means that a customer may buy “e” before “f” or “f” before “e”.

To find sequential rules, two measures are generally used: the **support** and the **confidence**. The **support** of a rule **X -> Y** is how many sequences contains the items from **X** followed by the items from **Y**. For example, the support of the rule {a} -> {e,f} is 3 sequences because {a} appears before the items from {e,f} in three sequences (seq1, seq2 and seq3).

The **confidence** of a rule **X -> Y** is the support of the rule divided by the number of sequences containing the items from X. It can be understood as the conditional probability P(Y|X). For example, the confidence of the rule {a} -> {e,f} is 1 (or 100 % if written as a precentage), because every time that a customer buy item "a", he then buy "e" and "f" in the example database. Another example is the rule {a} -> {b}. This rule has a support of 2 sequences and a confidence of 0.66 (that is 66%).

A **sequential rule mining algorithm** such as **RuleGrowth, ERMiner** and **CMRules** will output all sequential rules having a support and a confidence respectively no less than some thresholds *minsup* and *minconf* set by the user. For example, consider again the example database and suppose that the user set minsup = 2 and minconf = 60%. The following rules are found by RuleGrowth:

```
{a,b,c} -> {e} support = 2 sequences confidence = 100 %
{a} -> {c,e,f} support = 2 sequences confidence = 66%
{a,b} -> {e,f} support = 3 sequences confidence = 100%
{b} -> {e,f} support = 3 sequences confidence = 75 %
{a} -> {e,f} support = 3 sequences confidence = 100%
{c} -> {e,f} support = 2 sequences confidence = 100%
{a} -> {b} support = 2 sequences confidence = 66%
```

These rules can be viewed as more interesting than sequential patterns since they give a measure of confidence that they will be followed. For example, it is very informative to know that some rules such as {c} -> {e,f} have a confidence of 100 %.

In the past, I have carried a study with my student to compare the prediction accuracy of sequential patterns and sequential rules. In that study, we found sequential rules can provide a much higher prediction accuracy than sequential patterns when the patterns are used for sequence prediction. The reason is that sequential rules consider the probability (confidence), while sequential patterns do not.

Extensions of the task of sequential rule mining

In the previous paragraphs, I have introduced the topic of sequential rule mining. But note there also exists several extensions of the problem of **sequential rule mining.** These extensions have been proposed to address specific needs. I will provide a brief overview of a few extensions.

- **Discovering the top-k sequential rules**. The idea is to discover the ***k* most frequent rules** in a dataset having at least a confidence no less than *minconf*. For example, a user may specify that he wants to find the top 1000 rules having a confidence of at least 75 %. Some algorithms for this task are **TopSeqRules** and **TNS.**
- **Discovering sequential rules with a window size constraint**. This algorithm let the user find rules of the form X -> Y where X and Y must be close to each other with respect to time. For example, a user may want to find rules appearing whithin three consecutive itemsets in sequences. This is interesting for example for analyzing sequence of web clicks. An algorithm for this task is **TRuleGrowth**.

- **Discovering high-utility sequential rules.** Another extension is to discover rules where items may be annotated with quantities in sequences and each item may have a unit profit. For example, we may have a sequence where a customer bought three breads, then two apples and two bottle of milk and these items may have some unit profit of 1$, 2$ and 1.50$. The goal of **high-utility sequential rule mining** is to find rules that generate a high profit and have a high confidence (**high-utility rules**). An algorithm for this task is HUSRM.

Open-source implementations and datasets

There exists several algorithms for **sequential rule mining** and **sequential pattern mining** that have been proposed. Java implementations of the state-of-the art algorithms are currently offered in my open-source data mining library named SPMF.

It offers several state-of-the-art **algorithms for sequential rule mining** such as ERMiner (2014), TNS (2013), RuleGrowth (2011), TopSeqRules (2011), and CMRules (2010). Besides, SPMF offers several **algorithms for sequential pattern mining** such as CM-SPADE (2014), VMSP (2014), LAPIN (2005) and PrefixSpan (2004). To our knowledge, **ERMiner** is the fastest sequential rule mining algorithm. But RuleGrowth is still quite fast and consumes less memory. You can try the above algorithms by going to the SPMF website. On the website, you will find instructions about how to run algorithms and some datasets on the dataset page.

Applications of sequential rule mining

Some example of applications of **sequential rule mining** are e-learning, manufacturing simulation, quality control, web page prefetching, anti-pattern detection in service based systems, embedded systems, alarm sequence analysis, restaurant recommendation. For example, here are a few papers describing such applications:

E-learning

Fournier-Viger, P., Faghihi, U., Nkambou, R., Mephu Nguifo, E.: CMRules: Mining Sequential Rules Common to Several Sequences. Knowledge-based Systems, Elsevier, 25(1): 63-76 (2012)

Toussaint, Ben-Manson, and Vanda Luengo. “Mining surgery phase-related sequential rules from vertebroplasty simulations traces.” Artificial Intelligence in Medicine. Springer International Publishing, 2015. 35-46.

Faghihi, Usef, Philippe Fournier-Viger, and Roger Nkambou. “CELTS: A Cognitive Tutoring Agent with Human-Like Learning Capabilities and Emotions.” Intelligent and Adaptive Educational-Learning Systems. Springer Berlin Heidelberg, 2013. 339-365.

Manufacturing simulation

Kamsu-Foguem, B., Rigal, F., Mauget, F.: Mining association rules for the quality improvement of the production process. Expert Systems and Applications 40(4), 1034-1045 (2012)

Quality control

Bogon, T., Timm, I. J., Lattner, A. D., Paraskevopoulos, D., Jessen, U., Schmitz, M., Wenzel, S., Spieckermann, S.: Towards Assisted Input and Output Data Analysis in Manufacturing Simulation: The EDASIM Approach. In: Proc. 2012 Winter Simulation Conference, pp. 257–269 (2012)

Web page prefetching

Fournier-Viger, P. Gueniche, T., Tseng, V.S.: Using Partially-Ordered Sequential Rules to Generate More Accurate Sequence Prediction. Proc. 8th International Conference on Advanced Data Mining and Applications, pp. 431-442, Springer (2012)

Anti-pattern detection in service based systems,

Nayrolles, M., Moha, N., Valtchev, P.: Improving SOA antipatterns detection in Service Based Systems by mining execution traces. In: Proc. 20th IEEE Working Conference on Reverse Engineering, pp. 321-330 (2013)

Embedded systems

Leneve, O., Berges, M., Noh, H. Y.: Exploring Sequential and Association Rule Mining for Pattern-based Energy Demand Characterization. In: Proc. 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings. ACM, pp. 1–2 (2013)

Alarm sequence analysis

Celebi, O.F., Zeydan, E., Ari, I., Ileri, O., Ergut, S.: Alarm Sequence Rule Mining Extended With A Time Confidence Parameter. In: Proc. 14th Industrial Conference on Data Mining (2014)

Ileri, Omer, and Salih Ergüt. “Alarm Sequence Rule Mining Extended With A Time Confidence Parameter.” (2014).

Recommendation

Jannach, Dietmar, and Simon Fischer. “Recommendation-based modeling support for data mining processes.” Proceedings of the 8th ACM Conference on Recommender systems. ACM, 2014.

Interestingly, the above work found that sequential rules found by CMRules provided better results than other compared patterns found using FPGrowth and other algorithms.

Jannach, D., Jugovac, M., & Lerche, L. (2015, March). Adaptive Recommendation-based Modeling Support for Data Analysis Workflows. In Proceedings of the 20th International Conference on Intelligent User Interfaces (pp. 252-262). ACM.

Restaurant recommendation

Han, M., Wang, Z., Yuan, J.: Mining Constraint Based Sequential Patterns and Rules on Restaurant Recommendation System. Journal of Computational Information Systems 9(10), 3901-3908 (2013)

Customer behavior analysis

Noughabi, Elham Akhond Zadeh, Amir Albadvi, and Behrouz Homayoun Far. "How Can We Explore Patterns of Customer Segments' Structural Changes? A Sequential Rule Mining Approach." Information Reuse and Integration (IRI), 2015 IEEE International Conference on. IEEE, 2015.

Conclusion

In this blog post, I have given an overview of the tasks of **sequential rule mining** and **sequential pattern mining**, which aims at discovering patterns in sequences. Hope that you enjoyed reading it 😉 For researchers, there are many possibility of research on this topic.

By the way, if you want to know more about the topic of sequential pattern mining, I wrote a good survey that is easy-to-read on this topic: A Survey of Sequential Pattern Mining. It gives a good introduction and overview of this topic.

Philippe Fournier-Viger is a professor of Computer Science and also the founder of the open-source data mining software SPMF, offering more than 80 data mining algorithms.

Related posts:

Brief report about the IEA AIE 2016 conference

How to review a research paper?

招收数据挖掘领域博士后，地点：中国，深圳

This entry was posted in Big data, Data Mining, Data science, Research and tagged data mining, frequent patterns, high utility, sequential rules. Bookmark the permalink.

23 Responses to *An Introduction to Sequential Rule Mining*

Pingback: *An introduction to frequent pattern mining - The Data Mining & Research Blog*

Fernando Paladini *says:*

2015-10-20 at 3:41 PM

Another awesome article, thank you again!

Reply

Pingback: *An introduction to the discovery of periodic patterns in data - The Data Mining & Research Blog*

Tho *says:*
2016-11-11 at 5:42 PM

Thanks for the post. However, I have a question: what are differences between sequence mining and rule mining ? They look similar, aren't they?

Reply

Philippe Fournier-Viger *says:*
2016-11-11 at 7:10 PM

Hello, Glad you like the post. Sometimes the terms may have different meaning. But for me "Sequence mining" is more general. It means to discover patterns in sequences. But it does not say what kind of patterns. In a sequence, we could find many kind of patterns such as "sequential patterns", "sequential rules", "periodic patterns", etc. So, sequential rule mining is one task that you could call sequence mining, but there are also other tasks. Besides, some people will also discover patterns in 1 sequence, while other will find patterns that are common to multiple sequences. There are in fact many variations of these tasks. For example, if you do sequential rule mining in a single sequence, it is usually called episode rule mining instead of sequential rule mining, but you could still call it sequence mining.

Reply

Yuhao Yang *says:*
2017-03-26 at 2:30 PM

Thanks for the great post. I got one question, often a user will interested in both the "frequent sequential patterns" and "sequential rules". Among "the sequential rule mining algorithm such as RuleGrowth, ERMiner and CMRules", is there an algorithm that can generate "sequential rule" from "frequent sequential patterns", just as we may extract association rule from frequent itemsets. Thanks a lot.

Reply

Philippe Fournier-Viger *says:*
2017-03-26 at 6:44 PM

Hello,
Thanks. Glad you like the post. There is some algorithms such as RuleGen by Zaki (2001) that extract sequential patterns and use them to generate sequential rules. They will find some rules of the form X–> Y where X and Y are sequential patterns. But in my opinion these rules are not very good in practice, from my experience, as they are too precise and thus not noise tolerant. For example, a rule A,b,c -> B,D,E,F would be seen as different than A,b,c –> B,D,F,E or ACB –> BDEF or any variations. But in practice they are probably the same rule. So this creates a lot of problems for prediction because these rules are seen as different and thus the confidence and support may not be calculated correctly because of that. In my TKDE paper and related ADMA paper about RuleGrowth, I have shown that it is better to use rules of the form X–> Y where X is unordered and Y is unordered, but X must appear before Y, if we want to do prediction. This kind of rule is more general, and can work better in practice for prediction. At least, that is the results that I have found for prediction for webpage prefetching with some real data.

So in my TKDE paper about RuleGrowth, and the papers about ERMiner and CMRules, I have used the definition which is not based on sequential patterns. They find the rules directly. For the type of rules found by these algorithms, it is more efficient to find the rules directly rather than to use sequential patterns.

By the way, sequential rule mining is a topic that has not been explored a lot. There are many possibilities for research. If you want more details, you can e-mail me directly and I can give you some suggestions.

Best regards,

Reply

Yuhao Yang *says:*
2017-03-27 at 12:36 PM

Thanks a lot for the quick reply. That helps a lot. I'll send you an email for further discussion.

Pingback: *Comprehensive Guide on Data Mining (and Data Mining Techniques)*

Pingback: *An Introduction to Sequential Pattern Mining - The Data Mining Blog*

Siva *says:*
2017-08-16 at 9:27 PM

Hello,

Thanks for a great article. This is what I was exactly looking for. As you have already mentioned in your paper, there was lack of opensource implementations for sequences mining. I am looking for Python based implementation for sequence rule mining to cluster similar behaviors and identify anomalies if a new sequence is detected.
Could you please let me know if you have any idea on the same?

Thanks – Siva

Reply

Philippe Fournier-Viger *says:*
2017-08-17 at 5:29 PM

Hello,
You are welcome. Glad you like the article. Looks like an interesting topic to use sequential rules for clustering!

I personally do not program in Python, so I am not familiar with the libraries that are available in that language but I did not hear about such libraries either. I think that the easiest would be to call the SPMF library from your Python code. Also, the SPMF library can be used from the command line. So it could be called easily from a Python program.

By the way, I remember that there is a researcher in Canada, who implemented my ERMiner algorithm for sequential rule mining in Python. Send me an e-mail at philfv8 AT yahoo.com and I will give you his e-mail. Maybe he can share the code with you.

Best regards,
Philippe

Reply

Jeff *says:*
2017-08-21 at 12:06 AM

Hello,

So after reading your post, I realize that sequential rule mining task is different than my intuition led me to believe. It seems like you lose some of the sequential nature inherent in the data with sequential rules.

For example, suppose you have a dataset where the sequential rule {a} -> {x,y} has 100% support (and hence, 100% confidence). This rule can be realized in the dataset in one of three ways:
S1 =
S2 =
S3 =
It could be the case that
sup(S1)=100%, conf(S1)=100%
sup(S2)=0%, conf(S2)=0%
sup(S3)=0%, conf(S3)=0%
and so we conclude that x must occur before y. On the other hand, if it is the case that
sup(S1)=33%, conf(S1)=33%
sup(S2)=33%, conf(S2)=33%
sup(S3)=33%, conf(S3)=33%
then the order of x and y appearing after a does not matter.

Does the task of mining patters of the from X -> Y where X and Y are sequences (e.g., ->) have a name?

Reply

Philippe Fournier-Viger *says:*
2017-08-21 at 12:30 AM

Hello Jeff,

Something appear missing in your comment. I cannot read the S1, S2 and S3 of your example. But I think that I understand your point, so I will provide some comment on that below.

Generally, a rule is of the form X –> Y where something called X must appear before something called Y. X is the antecedent and Y the consequent.
Now, to be more specific, there are two main types of sequential rules in the literature:
– There is the one that I call "**standard sequential rules**" (SSR), where X is a sequence and Y is a sequence. For example, a rule a,b,c –> d,e means that a is followed by b, which is followed by b, and then c, which implies that it will be followed by d, and then e.
– There is another type of rules that I call "**partially ordered sequential rules**" (POSR). In these rules X is an unordered set of items, and Y is also an unordered set of items. But items in X must appear before those of Y. Thus, a rule a,b,c –> d,e, here means that a,b,c can appear in any order, but are then followed by d and e, in any order.

In my ADMA 2012 paper, I did some experiment to compare POSR and SSR for the task of webpage prediction to see which one is better:

Fournier-Viger, P. Gueniche, T., Tseng, V.S. (2012). Using Partially-Ordered Sequential Rules to Generate More Accurate Sequence Prediction. Proc. 8th Intern. Conference on Advanced Data Mining and Applications (ADMA 2012), Springer LNAI 7713, pp. 431-442.

It was found in that study that the POSR were more accurate than the SSR. The difference was sometimes more than 20 % more accuracy using the POSR. The reason is that SSR rules are too specific. A very small difference in the order of items will imply that several rules will be seen as different and their support or confidence may thus be

calculated incorrectly. For example, the rules:
a,b,c -> d,e
a,c,b –> d,e
b,c,a –> d,e
b,a,c –> d,e
a,b,c –> e,d
a,c,b, –> e,d
b,c,a, –> e,d
b,a,c, –> e,d
…
and so on would all be seen as different if we use the definition of SSR.
But in real life, these rules may actually represent the same thing but be slightly different because of noise or small variations. Considering these rules as different will greatly decrease the support of each of these rules, and may result in dozen of rules having a low support, and these rules may have confidence values that are largely different although they may represent the same thing.
If instead we use the POSR, we remove some constraint on the order and get a single more general rules instead of a dozen rules:
a,b,c -> d,e
This rules replaces all the above rules. Thus, it will also reduce the number of rules found. Actually, in pattern mining, a problem is always that there can be millions of patterns. So in my opinion it is good to generalize all these specific rules as one rule. This can greatly reduce the number of patterns to analyze. This rule will also be more noise tolerant.

I have also argued about that in my TKDE paper about POSR. You can read the two first pages of the introduction that argue that POSR are more appropriate then SSR:

Fournier-Viger, P., Wu, C.-W., Tseng, V.S., Cao, L., Nkambou, R. (2015). Mining Partially-Ordered Sequential Rules Common to Multiple Sequences. IEEE Transactions on Knowledge and Data Engineering (TKDE), 27(8): 2203-2216.

Also, to my knowledge, POSR have more applications. They have been used in several papers, while I did not see much about SSR for applications.

But of course, SSR could also be more appropriate than POSR for some domains, including what you are doing! There is never something that is always best. And all of these definitions could be modified for some specific applications if other constraints would need to be handled.

Best,

Philippe

Reply

Jeff *says:*
2017-08-21 at 12:16 AM

Hello,

So after reading your post, I realize that sequential rule mining task is different than my intuition led me to believe. It seems like you lose some of the sequential nature inherent in the data with sequential rules.

For example, suppose you have a dataset where the sequential rule {a} -> {x,y} has 100% support (and hence, 100% confidence). This rule can be realized in the dataset in one of three ways:

S1 = <{a},{x},{y}>
S2 = <{a},{y},{x}>
S3 = <{a},{x,y}>

It could be the case that

sup(S1)=100%, conf(S1)=100%
sup(S2)=0%, conf(S2)=0%
sup(S3)=0%, conf(S3)=0%

and so we conclude that x must occur before y. On the other hand, if it is the case that

sup(S1)=33%, conf(S1)=33%
sup(S2)=33%, conf(S2)=33%
sup(S3)=33%, conf(S3)=33%

then the order of x and y appearing after a does not matter.

Does the task of mining patters of the from X -> Y where X and Y are sequences (e.g., <{a}> -> <{x},{y}>) have a name?

Reply

James *says:*
2017-09-08 at 11:45 PM

original
“consider again the example database and suppose that the user set minsup = 0.5 and minconf = 60%.”

FIXED
“consider again the example database and suppose that the user set minsup = 2 and minconf = 60%.”

Reply

Philippe Fournier-Viger *says:*
2017-09-09 at 12:08 AM

Hi James,

Glad to see you are still reading on the blog. Thanks again for finding these errors 🙂 Very helpful.

Best,
Philippe

Reply

Pingback: *How to discover interesting patterns in data? - The Data Mining Blog*

Simon Kang *says:*
2018-02-11 at 8:17 PM

Thank you for the post. Reading the post, I started wondering how the prediction part is done. Could you give a brief explanation about how the sequential pattern prediction is done?

Reply

Philippe Fournier-Viger *says:*
2018-02-11 at 11:11 PM

Hi, Glad you like the post. For the prediction using sequential rules, the main idea is not very complicated.
I will explain this with an example. Let's say that we have sequences of webpages visited by some users on a website. Having these training sequences, we can extract some sequential rules from them. We could find a rule such as: {a} –> {b} support: 50% confidence: 90 % This means that when a user visit the webpage {a} 90 % of the time, he will visit the webpage {b} after.
Then, let say that we have a new user who visits the following webpages in that order e,c,a Then, we can try to match the different rules that we have with that sequence to predict what the user will visit next. To do so, we compare each rule with the sequence to see which rule can be used to make a prediction. For example, we can find that the rule {a} –> {b} match with that sequence because the left side of the rule "a" appears in e,c,a. Thus we can predict that the right side of the rule {a}–> {b} will be the next page visited by the user, that is the webpage {b}.
If several rules match with a sequence, we can use the support and confidence to decide which prediction is more likely, or we can add other criteria. This is the basic idea. We have some training sequences. We extract some rules. Then we try to match the rules with th sequences to see what will happen next.

If you want more details, you can read my papers about using sequential rules for making prediction. It explains this in more details and show some experiments with webpage prediction:
PDF: http://www.philippe-fournier-viger.com/sequential_rules_prediction_2012.pdf

Best regards,

Reply

Leo *says:*
2018-05-25 at 4:53 PM

Hello, first of all thank you for your application and all related explanations, it's very useful. I have a question: is there an algorithm which keep items order in rule generation?

In post example we will have:
{a,b,c} -> {e} support = 1
{a,c,b} -> {e} support = 1
{a,b} -> {f,e} support = 2
{a,b} -> {e,f} support = 1

Thanks.

Reply

Philippe Fournier-Viger *says:*
2018-05-25 at 7:19 PM

Hello,
Yes, there exists some like RuleGen, which is implemented in SPMF, and others which I have not implemented. But an issue with this kind of rules is that many rules that are only slightly different will be viewed as different rules. Thus, it will decrease the support of all the rules, and some very similar rules may have very different support and confidence. But maybe that it is what you need for your application.

Best,

Reply

Pingback: *Introduction to time series mining with SPMF - The Data Mining BlogThe Data Mining Blog*