Data Mining by Non-Axiomatic Reasoning

Patrick Hammer[1]

Computer & Information Sciences Department
Temple University, 347 SERC, 1925 N. 12th Street
Philadelphia, PA 19122, USA
Email: `patrick.hammer@temple.edu`

Abstract. Data Mining takes many forms and is usually carried out by algorithms that were specially designed for the task at hand. In this paper we present the usage of a Non-Axiomatic Reasoning System (NARS) for Data Mining purposes. NARS is a general-purpose reasoner that enables reasoning about data while not being restricted to specific Data Mining problems, allowing for a broad range of applications. This paper evaluates the approach on specific data sets, and contains improvements and comparisons with other techniques. Additionally, the included LazyMiner use case demonstrates autonomous NARS-based rule mining under hard resource constraints on mobile devices.

Keywords: Data Mining, Non-Axiomatic Reasoning System, Data Stream Mining, Rule Mining, Data Mining by Reasoning

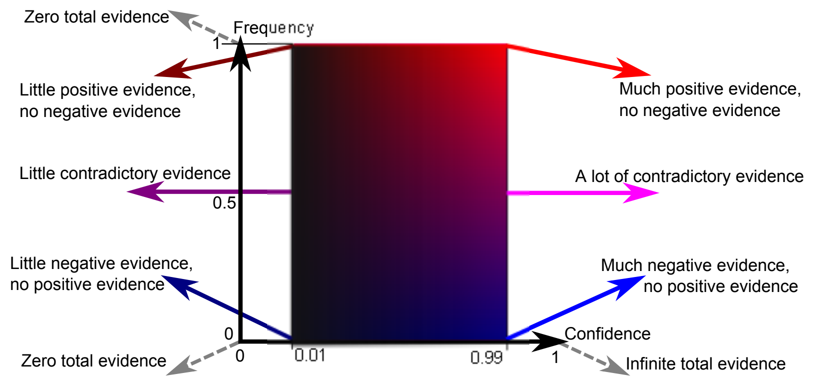
1 Introduction

Due to the diverse nature of different Data Mining problems, Data Mining takes many forms and is usually carried out by algorithms that were specially designed for the type of problems they are applied on. This involves the usage of specific data structures, for internal knowledge representation, together with efficient program flow to effectively extract desired information from the data. Additionally, algorithm-dependent parameter-tuning, data preparation and feature selection, to various degree, is usually expected to be done by the user. This is partly in contrast to recent developments, such as in Deep Learning, where feature selection and feature engineering, are to a large degree obsolete. Similarly, as striving for higher level of abstractions in programming improves programs and readability, achieving a higher level of generality in Data Mining techniques can make the application of these approaches easier and lead to better results.

In the extreme case of generality lies the grand dream of "Artificial General Intelligence" (see [1]), systems that learn by their own experience, don't depend on labeled data, feature selection or parameter tuning, and which can solve a large variety of problems in different domains autonomously. However, although a significant body of research exists in this area, only a few promising prototypes of such systems exist. One such system, NARS (Non-Axiomatic Reasoning System), has reached the necessary level of maturity for a comparison with other data mining techniques, and for potential applications. This paper includes evaluations and comparisons to other data mining techniques, improvements to NARS, as well as an representative application for rule mining, “LazyMiner”, under hard resource constraints on mobile devices.

NARS is a general-purpose reasoning system with the ability to learn from its experience and to work with insufficient knowledge and resources. It does not depend on labeled data, works in real-time (new input will be accepted at any time) and is designed to uniformly explain and reproduce many cognitive functions, including perceiving, planning, reasoning, learning, and so on. The logic it uses is called NAL (Non-Axiomatic Logic). It is a non-boolean term logic that allows for conclusions with various degree of certainty. NAL also includes a formalization of evidence that allows the system to keep track of the collected evidence that speaks for or against a hypothesis/statement (see [2]). Open-NARS (see [3]) is the most mature implementation of NARS, which is the version that is improved and evaluated in this paper.

Since NARS uses a term logic, all input is of this form. This term logic usually consists of $(A \rightarrow B)$ statements which represent an Inheritance relation between A and B. For instance, that cats are animals can be written as $(cat \rightarrow animal)$. A further relation is Implication. Implication, $\Rightarrow$, is taken as a correlative rather than causal relation, and can be used to encode a causal relation as a special case, see [4] for why this is the case. Important to note is that the truth value of statements is not boolean, rather it is a tuple of the positive evidence w_+, and the negative evidence w_-, that votes for/against the statement. Additionally, the total evidence of a statement is intuitively defined as $w := w_+ + w_-$. An alternative representation we will use from now on is defined as the tuple (f, c) with frequency $f := \frac{w_+}{w}$ and confidence $c := \frac{w}{w+1}$. This representation can be mapped to the former more intuitive representation in a bi-directional manner, the following picture illustrates this:


Zero total evidence
Frequency
1
Little positive evidence, no negative evidence
Much positive evidence, no negative evidence
Little contradictory evidence
A lot of contradictory evidence
0.5
Little negative evidence, no positive evidence
Much negative evidence, no positive evidence
0
Confidence
Zero total evidence
0
0.01
0.99
1
Infinite total evidence

To form such statements, the system can form compound events such as sequences of events. In sequences, the relative time information between the involved events is retained. Additionally, Temporal Induction, an Induction rule, allows the system to create predictive statements based on two events. The induction rule basically states

$$\{A(f_1, c_1), B(f_2, c_2)\} \vdash (A \not\Rightarrow B)(f_1, \frac{f_2 * c_1 * c_2}{f_2 * c_1 * c_2 + 1})$$

where event A happened before B. The result can then be revised on each occurrence by summing up the positive and negative evidence, allowing stronger and stronger hypotheses to form. Here, in the simplest case, A and B are events, where usually both are Inheritance statements such as $(weather \rightarrow [rainy])$ and $(street \rightarrow [wet])$ (brackets denoting properties, see [2]), but they could also be sequences.

To decide which premises should be selected for inference, a control mechanism is necessary. In this paper, only the aspects of the NARS control mechanism relevant to Data Mining are described, more details can be found in [3]. NARS's control mechanism realizes a form of Attentional Control using the Bag data structure. This is a data structure where the elements are sorted according to their priority, and the sampling operation chooses candidates with selection chance proportional to their priority. This makes the control strategy similar to the Parallel Terraced Scan in [5], as it also allows to explore many possible options in parallel, with more computation devoted to options which are identified as being more promising. After the selection, a candidate is returned to the bag, with a decrease in priority proportional to the durability value. Data items enter having their priority modulated by their truth value and occurrence time. This is the case for both, input events and results generated by the inference process. All combinations of premises happen through sampling from a bag. Both recent and high truth-value items tend to be used more often, but also query-related ones. This suffices to allow for simple attentional control, which chooses premises neither fully randomly nor exhaustively, but biased by the relevance of patterns and the current time. This is helpful as each inference step is valuable: only a fixed number of inference steps can happen per time unit, so how to utilize them matters a lot.

LazyMiner is our representative application for rule mining under hard resource constraints using Non-Axiomatic Reasoning. Rule mining plus inference allows for enhancements of existing applications but also for new possibilities: learned and inferred knowledge can reduce sensing demands and lower energy consumption, can give insights about user habits, and can be used to provide users with contextually relevant information. Such technology can support and complement a large variety of energy-efficient and context-aware applications. It opens the door to all sorts of applications that need to extract and work with relationships between events. However, allowing for efficient rule mining directly on mobile devices is a challenge. Software such as [6] and [7] circumvent that problem by letting the rule mining process happen on a server. LazyMiner is an attempt to get rid of this restriction by incremental rule mining with user-defined resource usage. LazyMiner makes directly use of OpenNARS, which is designed to work under the Assumption of Insufficient Knowledge and Resources (AIKR). Applying this system allows for precise control of how many resources should be assigned to the rule mining process, making it tractable to be applied directly on mobile devices. Here, the rule mining happens through the application of temporal inference rules, which allow temporal patterns, from event streams, to be summarized. In such event streams, the amount of possible event combinations grows exponentially with the amount of incoming events. Thus, exhaustively evaluating all possible combinations is extremely expensive and often not an option. Instead, the Attentional Control mechanism of NARS is utilized, which allows it to allocate resources on patterns that seem promising, stable and useful, while keeping the overall resource usage of the system constant.

2 Related work

General AI systems, which are also capable of extracting knowledge from data, such as AERA (see [8]) and OpenCog (see [9]) exist.

AERA follows similar principles to NARS, especially in its Resource Allocation strategy that shares strong similarity with the Bag-based mechanism in NARS. It is also similar to the Parallel Terraced Scan proposed in [5]. Roughly speaking, all of them allow for the parallel execution of “code items” with priority-controlled speed. However, AERA’s learning component is not yet fully implemented.

OpenCog, although also making use of a Term Logic (and others), follows different principles. Especially it does not treat Resource Allocation for working with insufficient resources as a fundamental requirement. However, this system is still under heavy development.

For particular Data Mining tasks the situation is much better, as Data Mining can be seen as a mature field with working solutions to many specific problems. One of many examples is the Generalized Sequential Pattern algorithm (GSP) and extensions thereof (see [10,11]) as a solution to the Sequential Pattern Mining problem. GSP is successfully applied in a variety of applications. Similarly, the Apriori algorithm, [12] and variations thereof, are applied in various applications that demand to mine for Association Patterns.

LazyMiner Regarding LazyMiner, there is related work in the literature, such as MobileMiner and the “Acquisitional Context Engine” (ACE), described in [7] and [6] respectively. MobileMiner also supports rule mining from data streams, but also prediction based on the rule mining results. However, their approach suffers from a key limitation: the rule mining happens in an exhaustive and non-incremental manner, meaning that the resource usage depends exponentially on the amount of input. To reduce resource usage, they make use of the ”Weighted Mining of Temporal Patterns” (WeMit) idea. This idea consists of grouping events within a certain chosen time window into the same baskets, and to represent re-occurring baskets as a single compressed basket. This treatment reduces the amount of possible basket combinations significantly, for all the cases where it applies. But even with this idea, their resource requirements grow exponentially in respect to the amount of resulting compressed baskets. In the worst case of temporally sufficiently far apart and distinct input events, the amount of resulting baskets is equal to the amount of input events itself. Thus, for certain scenarios, the rule mining becomes too expensive to be directly carried out on mobile devices. On the other hand, LazyMiner, using NARS, mines patterns incrementally and attention-driven, while keeping the total resource usage within a pre-defined constant. In ACE (see [6]), rule mining works in an incremental manner, and is targeted at energy-efficient context sensing. It also allows for boolean inference based on the rule mining results. While for this approach it is less clear whether it could be made efficient enough to run on a smart-phone, its limitations to only include minConf=99% results for its reasoning, makes it inapplicable for many cases: it misses rules that most time apply, but not always. And because of this it also misses potential predictions and inference results, making it effectively fail in less stable environments. For LazyMiner the situation is different, because it makes use of Non-Axiomatic Logic (NAL) (see [2]) that allows for conclusions with variable certainty, dependent on the certainty of the premises.

3 Methodology

Data Mining and Reasoning are often seen as orthogonal to each other: While Data Mining is all about extracting knowledge from data, Reasoning systems usually completely lack the ability to learn anything from data. Instead, they usually derive conclusions which logically follow from the premises. Not so NARS, which has a notion of certainty and several "structure building" rules. Together with its general knowledge representation language, this helps it to be applicable to various data mining problems. Additionally, Reasoning Systems usually have a general knowledge representation language which is not restricted to any problem domain, allowing NARS to answer different questions about the data. Also the inference rules as well as the control mechanism are usually completely domain-independent. This flexibility and domain-independence was also a major consideration for designing NARS as an inference engine (see [2]), and a key motivation for us to apply it to different Data Mining problems.

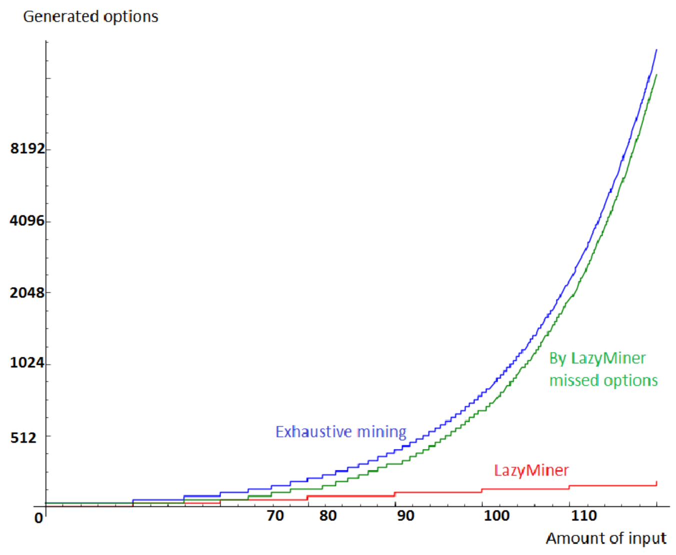
To evaluate the strength of the system, but also to identify and fix weaknesses, the system was compared with other methods, on different data mining problems. To obtain results of statistical significance, the experiments were repeated multiple times, using different data samples. This allowed us to see general trends and to define success rate measurements. In this process we also identified weaknesses, and found ways to improve the system.

4 Results

Our results from the comparisons:

1. Time series data: We wanted to get an understanding of how working under AIKR affects the systems pattern mining capability. Here, a transformation to Sequential data was carried out, achieved by a Discretization technique, as also suggested as an option in [10]. To achieve this, all values in the

used data set [13] were normalized into the range from 0 to 1 and rounded towards the closest point in $\{0.1 * k | k \in \{0,1,2,3,4,5,6,7,8,9,10\}\}$ such that the maximum value in the data set became 1.0 and the minimum value 0.0 and the others in-between. In this way the problem was reduced to the Sequential pattern mining problem for which an extension of GSP (see [11]) is a solution for. GSP constructs patterns incrementally and makes use of the Apriori principle to prune the tree. However, if we set a low $minSupp$ and $minConfidence$, there are still exponentially many patterns in respect to the input amount which are above $minSupp$, so if directly used that way it behaves like an exhaustive pattern miner. Such an exhaustive GSP-based miner becomes an upper bound to compare with, as by definition there is no way for NARS to find a pattern that is not also found by an exhaustive approach. This way, the amount of patterns found by NARS, compared with the amount of total possible patterns, gives an indication of overall capability under the chosen resources. We obtained across 100 runs, using an input stream of a fixed amount of input events grouped into buckets of 10 (from the [13] data set) per time unit:



The picture suggests that the bucket combinations missed by NARS grow exponentially in respect to the amount of input events seen so far, meaning that the system misses most of the options.
The red line can be easily explained, as the amount of processing NARS can do from one time unit to the next is a constant. So the amount of new patterns it will derive will, at most, linearly increase over time even though a constant amount of new events enters the system per time unit.
For the blue curve it is easy to see why it is really growing exponentially: for a list of events $a_1, ..., a_n$ and a new incoming event a_{n+1}, at least the patterns $a_1, ..., a_n$ and $(a_1, a_{n+1}), ..., (a_n, a_{n+1})$ are both possible to extract,

so there are at least twice as many candidate combinations of events after a new event is inserted.

Putting these considerations together, with the green function being the red at most linear function subtracted from the blue exponential one, we indeed end up with an exponential function.

And how could it have been otherwise? The resource expense, to generate all options, grows exponentially with each new input event, while NARS invests the same constant amount of time on each one. Even when more resources are assigned to NARS, the slope of the red curve will be steeper, but the result will be the same. However this is not bad considering that most patterns appear locally in time. This kind of pattern is clearly also seen by NARS, as priorities of recent events will be higher and thus have a higher chance to be selected together.

2. Data Stream Mining: While NARS treats the Sequential Pattern Mining problem as a Data Stream Mining problem by default, GSP can also be applied together with Reservoir sampling. (see [14]) This makes it possible to keep the computational cost under control as the reservoir always holds a certain constant amount of items similar as Bag. Reservoir sampling however keeps all input events with equal probability, so doesn't have the aforementioned Bag-sampling property to prefer temporally local patterns. This makes it less useful for this purpose, and also unable to deal with concept drift properly. On the other hand it more easily keeps track of the frequent patterns, which suggested an improvement to Bag-based sampling to us: increasing the "quality" of an event whenever it occurs, a value that represents a threshold under which priority can only slowly drop.

LazyMiner Application LazyMiner addresses the issue of high computational expense in rule mining processes on mobile devices. Having NARS as the core component allows LazyMiner resources limits to be set using an API call. This makes it tractable to be used on mobile devices. One goal of the project is to show that the most useful results are found by the Bag-based Attentional Control mechanism, even when low resources are provided and a high number of input events are entered. This is demonstrated by collecting statistics of cases where frequently re-occurring patterns are found while operating under such conditions. Such cases consist of re-occurring sequences of events, with 'unrelated' events potentially in-between. Also reasoning and question answering capabilities are shown based on such examples. Here, the events usually incorporate data from all sorts of sensory devices. Being able to use sensory input directly is an additional goal of the LazyMiner application. Several input encoders that map typical sensory data to statements in Non-Axiomatic Logic (NAL) were developed for this purpose.

In our experiments, a LG Arist MS210 device was used, with Android 7.0 as operating system. LazyMiner runs with Android 4.4 (using the ART runtime instead of Dalvik) and newer. OpenNARS is instantiated a single time and is controlled by LazyMiner. LazyMiner comes with a convenience API and a variety of encoders that allow for an easy way to input all sorts of sensory events into the NARS instance. The encoders automatically map sensory data to the format the NARS instance accepts.

An issue to address was how to encode sensory values into a format acceptable to NARS. Here, the simplest way of encoding values into terms turned out to be discretization. Using this method, a term is assigned to a certain numeric range. This makes it possible for the system to explicitly reason about the involved input quantities. The result of such a quantization is usually an event like

```
<{50} --> heartrate>.
```

This is the representation[1] the default sensory value encoders in LazyMiner use. Additionally there is a mode for completely preserving the numeric quantities by modulating the truth value (frequency) of the statement. This idea works when no terms for intermediate values are necessary. This isn't the case in general, although, for example, it makes sense to encode the brightness value of a pixel as a statement in such a way that, its truth frequency corresponds to its brightness. This leads to a representation like

```
<{pixel1} --> [bright]>. %degree%
```

where degree is the truth frequency of the statement. The distinction between both methods is related to brightness vs. color vision in human beings, a topic that is beyond the scope of this report.

To run OpenNARS on mobile devices, OpenNARS was ported to Android so that it compiled successfully. Porting the Java8 code to the Android platform only required minor changes to the code.

The API was then refined, to make it possible to feed NAL statements into the reasoner and to register callback functions for reasoning results. A simpler user interface, that allows the monitoring of the reasoner was also developed.

Also, a LazyMiner-specific API was added, as application designers do not want to deal with the details of NARS when applying LazyMiner, this API acts as an abstraction layer. It encodes a large range of device-specific events and other custom events in a format NARS accepts, for instance it can deal with numeric array inputs. Additionally, it allows resource limits to be defined. And the developed GUI allows for convenient usage and general monitoring of the reasoner.

Then, the following interactive application in the domain of health care was developed. It shows different abilities of LazyMiner, such as learning from sensory data and other events, as well as answering questions about learned knowledge.

Patient example To evaluate the system in a more realistic way, whilst making our results reproducible, instead of using a fixed data set, we created a Patient data generator (see [15]) that simulates a typical daily routine of the patient. It involves the simulation of the patient's heart rate in regard to different activities as well as some random fluctuations. The heart rate sensor constantly inputs numeric events such as

[1] As the notation suggests, it is the ASCII version of $(\{50\} \rightarrow \mathit{heartrate})$

```
<{60} --> heartrate>.
```

into the system, where the value was originally between 0 and 100 and was rounded to the next 10 for discretization. Additionally, location information and activity information, such as whether the person is running, sleeping or walking, is provided to LazyMiner. Location information is encoded as an

```
<(*,{SELF},location) --> at>.
```

event, and activities are encoded by operations like

```
<(*,{SELF}) --> ^run>.
```

Using these events, the system can be queried as to answer why the heart rate sometimes suddenly raises to a frequency of 110. In the simulation, this value is only reached after the patient is eating at home, potentially because something poisonous was eaten.

LazyMiner, using NARS, showed competence in finding "eating at home" as an explanation of the increased heart-rate: after simulating 7 complete day cycles using the data generator, for 100 times, the explanation was found in 93 of the cases, which also means 93 percent of overall success rate.

5 Discussion

NARS enables general data mining with constant computational resources, and our findings suggest how such a system can be further improved. Specifically, a Long-term importance value, increased by pattern frequency, can lead to frequent pattern mining improvements for Bag-based methods. As we have shown, NARS is not the ideal choice when all possibilities can be evaluated exhaustively, in this case an approach like GSP is usually better. But it presents another technique for data stream mining, where exhaustive data mining is usually infeasible. Here it has the benefit to extract local patterns more easily over the approach that combines Reservoir Sampling with GSP. This allows it to deal with concept drift more effectively, which is especially important for autonomous systems.

With our LazyMiner application, we have also shown that the system allows for energy-efficient open-ended rule mining on mobile devices, and can support context-aware applications that can benefit from its usage. Furthermore, our example in the health care domain shows that the system is capable of finding answers to relevant queries, such as to find reasonable explanations for events of interest. We also expect similar capabilities also in different domains, whenever extracting knowledge from event streams is important.

Whilst the project goals are met, applying Non-Axiomatic Reasoning systems to such tasks will be further explored in the future. Also further comparisons with other methods will be necessary, to obtain a more complete picture of the system's data mining capabilities. The code of the project will be publicly available on Github, see [16].

Acknowledgements

Special thanks to Dr. Pei Wang and Tony Lofthouse, for their suggestions on how NARS could be applied to these problems.

References

1. P. Wang, B. Goertzel, and S. Franklin, "Artificial general intelligence," *Proceedings of the First Artificial General Intelligence Conference, IOS Press*, vol. 172.
2. P. Wang, "Non-axiomatic logic: A model of intelligent reasoning," *World Scientific*, 2013.
3. P. Hammer, T. Lofthouse, and P. Wang, "The opennars implementation of the non-axiomatic reasoning system," *Artificial General Intelligence, Springer, Cham*, pp. 160–170, 2016.
4. P. Wang and P. Hammer, "Issues in temporal and causal inference," *Artificial General Intelligence, Springer, Cham*, pp. 208–217, 2015.
5. J. Rehling and D. Hofstadter, "The parallel terraced scan: An optimization for an agent-oriented architecture," *Intelligent Processing Systems, IEEE International Conference*, vol. 1, pp. 900–904, 1997.
6. N. Suman, "Ace: exploiting correlation for energy-efficient and continuous context sensing," *Proceedings of the 10th international conference on Mobile systems*, pp. 29–42, 2012.
7. S. Vijay, S. Moghaddam, A. Mukherji, K. K. Rachuri, C. Xu, and E. M. Tapia, "Mobileminer: Mining your frequent patterns on your phone," *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 389–400, 2014.
8. E. Nivel and C. T. et. al., "Autocatalytic endogenous reflective architecture," *ISSN 1670-5777*, 2013.
9. D. Hart and B. Goertzel, "Opencog: A software framework for integrative artificial general intelligence." *Proceedings of the First Artificial General Intelligence Conference, IOS Press*, pp. 468–472, 2008.
10. C. Aggarwal, "Data mining, the textbook," *Springer, ISBN 978-3-319-14142-8*, 2015.
11. Y. Hirate and H. Yamana, "Generalized sequential pattern mining with item intervals." *JCP*, pp. 51–60, 2006.
12. R. Agarwal and R. Srikant, "Fast algorithms for mining association rules." *Proceedings of the 20th VLDB Conference*, vol. 1215, pp. 487–499, 1993.
13. "Activity recognition from single chest-mounted accelerometer data set." [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Activity+Recognition+from+Single+Chest-Mounted+Accelerometer
14. B. Babcock, D. Mayur, and R. Motwani, "Sampling from a moving window over streaming data." *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 633–634, 2002.
15. "The patient data generator." [Online]. Available: https://gist.github.com/patham9/b9bc59d6ece98986e821b1c7cff96018
16. "Lazyminer repository." [Online]. Available: https://github.com/patham9/lazyminer