Non-Axiomatic Reasoning Systems as a Substitute for Recommender Systems

https://github.com/AllHailTheSheep/Non-Axiomatic-Reccomendation-System

Julia Fasick

CIS 4360: Artificial General Intelligence

Pei Wang

May 2, 2025

Introduction to the Problem

In the modern world, consumers are constantly bombarded with recommendations. Social media, streaming services, ad platforms, and news outlets all use recommender systems to allow users to efficiently navigate content suited to their interests. In recent years, with the mass adoption of social media apps such as TikTok and Instagram, recommender systems have become more of a feature than ever before. While previous recommendation systems sat in the background, with the user none the wiser, modern algorithms play a tangible part in the user's experience. With the ever-increasing importance of these systems, they have become more and more refined, often combining multiple types of content filtering into one recommendation system (Kalideen & Yağlı, 2025). As the systems get more complicated, so does the power needed to run them (Roy & Dutta, 2022). Meta's dacenters accounted for 5,141,350 metric tons of C02 scope 2 equivalents in 2023, and consumed 14,975 GWh of electricity (Meta, 2024). Additionally, traditional recommendation systems suffer from several other issues. "The cold-start problem" is the tendency of traditional recommender systems to perform very poorly initially, as they do not have enough data about user preferences to draw similarities to other liked items, users, etc. Data sparsity is also an issue, which occurs when there are not enough similar data points and the system has to reach for more and more dissimilar points. In addition, some recommendation systems suffer from overfitting, which results in no new content being recommended to the user and preventing the system from learning the user's newer preferences. There are several other issues, such as the gray sheep problem, latency issues, and synonym issues (Kalideen & Yağlı, 2025), but they are not the focus of this paper. These problems can be mitigated by using non-axiomatic reasoning systems and moving content filtering from the lens of machine learning to that of artificial general intelligence. This paper aims to provide a starting

point for creating more efficient and tailored recommendation systems using a non-axiomatic reasoning system (NARS). The example we will use will be an outfit recommendation algorithm. A user can specify how much they like a specific piece of clothing, and the system will recommend an outfit for the day based on characteristics such as temperature, UV index, whether or not it is raining, and user specific preferences. The preferences can include a user saying “I do not care to be fashionable today”, which would lead to the system producing an outfit more aptly suited for comfort than looks. There are several advantages to this system. Primarily, the system can use a system similar to fuzzy logic to find the best fit for its parameters. In a traditional system, there would be a hard cutoff, where, for example, above a certain temperature no warm outfit would ever be chosen. Using a non-axiomatic reasoning system, there is no such hard cutoff. The outfit selected will be based on user preferences and learned behavior as opposed to hard limits. Additionally, it hopes to be a more efficient and scalable solution. By only abstracting only the concepts it needs to, NARS can draw conclusions based on evidence, as compared to a set of hard parameters (Wang, 2025).

Introduction to NARS

First, to use the system we must learn its language. We will not go into too much depth, but a brief overview is necessary. The language of NARS is Narsese, and it mimics English in many ways. Individual terms are abstracted by their similarities to other terms or concepts. The most basic copula, or relation, between terms is the inheritance copula, represented by “-->” in typeface and often seen as →. There are many other copulas derived from the inheritance copula, including use-case specific custom copulas, but the main one we will use is the inheritance copula. For example, the idea that “sheep is a type of farm animal” could be represented in Narsese as <sheep → farm-animal>. In this example, sheep is the subject term and farm-animal

is the predicate term. The statement can be followed by two values to influence how NARS draws conclusions from the evidence given. These two values are *f* and *c*, for frequency and certainty. The first, *f*, is frequency, or the ratio of positive and negative evidence for such a judgement. The second, *c*, is confidence, or the certainty in the frequency as the truthful proportion of evidence (Wang, 2025). A statement with a high confidence will be less altered by new evidence received by the system, even if it drastically contrasts what the system already believes. For example, <ostrich → farm-animal>. %0.5;0.25% would mean that we are 50% certain that an ostrich is a farm animal and 25% confident in our answer. If more evidence was submitted to the system, it would affect the truth value pair a significant amount as we are not confident in our answer. These dual numerical uncertainty values allow for "degrees of uncertainty" which allow NARS to draw conclusions from insufficient evidence (Wang, 2011). To represent a proper noun, we can surround a term in brackets. <{Snoopy} → dog>. would tell the system that Snoopy is an instance of dog. This is called an extensional set of dog, that is, Snoopy is an extension of dog. To specify a property of something, we use brackets. <dog → [lazy]>. signifies that playful is a property of dog, which would be known as the intensional set of dog. With our previous example as well, NARS will draw the conclusion <{Snoopy} → [lazy]>. This conclusion will have a truth value associated with it lower than either of the judgements used to reach it (Wang & Awan, 2011).

There are many ways NARS will draw conclusions. These include (but are not limited to) deduction, induction, abduction, and revision. Deduction is simply how the inheritance copula can be extended between multiple statements. With the above examples about Snoopy, with the addition of the statement <dog → animal>. NARS will deduce that <{Snoopy} → animal>. Induction uses common instances in the extensional set to draw a conclusion. Given the

statements <{Snoopy} → animal>. and <{Snoopy} → dog>. NARS will induce that <dog → animal>. Abduction, in comparison, uses common properties in the intensional set to draw conclusions. Given that <{Snoopy} → [lazy]>. and <dog → [lazy]>. NARS will abduce that <{Snoopy} → dog>. Induction and abduction are weaker conclusions, and will result in lower frequency and conclusion values deduction. The revision rule simply summarizes the evidence available to the system. If <{Snoopy} → dog>. is derived separately from two separate rules, the revision rule will summarize the evidence available into one concrete judgement. The figure below displays the calculations of f and c for each syllogistic rule. Note the inclusion of k in some of the equations. k is the personality parameter or evidential horizon, that is, the amount new evidence is considered in the system. k =1 is a standard choice and will be used for our purposes.

Rule	f	c
Deduction	$f = f_1 f_2$	$c = f_1 f_2 c_1 c_2$
Induction	$f = f_1$	$c = (f_2 c_1 c_2)/(f_2 c_1 c_2 + k)$
Abduction	$f = f_2$	$c = (f_1 c_1 c_2)/(f_1 c_1 c_2 + k)$
Revision	$f = 1$	$c = (f_1 f_2 c_1 c_2)/(f_1 f_2 c_1 c_2 + k)$

(Wang, 2025)

We can also group multiple predicates in one statement by using &. <{Snoopy} → (&, [lazy], dog)> would tell the system that Snoopy is an instance of dog and that Snoopy has the property of being lazy. This is especially useful when asking NARS questions. We can ask the

system a question by using ?x as a variable. <?x → (&, [lazy], dog)? would ask the system what is a lazy dog, and would answer Snoopy.

A Non-Axiomatic Recommendation System

Our example of the recommendation process will be an outfit recommendation system. Given several articles of clothing, the system will recommend an outfit based on the user's preferences and the overall parameters of the day. The first step is to abstract our product. This means we must describe our products and their qualities. We will use three properties to describe our articles of clothing: warmth, fashionability, and how much the user likes them. For each article of clothing, we will define it as a type before defining its properties. An example of how jeans are defined is as follows:

<{jeans} --> bottom>. %1.00;1.00%

<{jeans} --> [warm]>. %0.75;0.9%

<{jeans} --> [fashionable]>. %0.75;0.9%

<{jeans} --> [liked]>. %0.9;0.9%

This allows us to then ask the system questions such as <?x --> (&, [fashionable], bottom)>? to find the bottoms that are most fashionable. One thing to take note of is that we are declaring jeans to be an instance of jeans, i.e, Jeans with a capital J. While the English translation at this point begins to fall apart, this helps the system understand the relationship between the instance of articles of clothing and the actual types of clothing they inherit. Another is that we are declaring the confidence of each observation at 0.9, except for the inheritance statement which gets 1 as it is set in stone. This is because NARS will revise the frequency of judgements. A value of 0.9 allows us to revise our initial judgment down the line based on user preferences while also still considering new evidence. The following table lists each article of clothing we represent and its given frequency for each property.

Article	Inherits	warm	fashionable	liked
snowboots	shoe	0.8	0.1	0.25
combatboots	shoe	0.5	.75	0.75
sneakers	shoe	0.25	0.5	0.8
heels	shoe	0.1	0.9	0.25
sweatpants	bottom	0.75	0.2	0.75
jeans	bottom	0.75	0.75	0.9
slacks	bottom	0.5	0.75	0.25
gymshorts	bottom	0.1	0.25	0.5
longsleeveshirt	top	0.75	0.5	0.75
shortsleeveshirt	top	0.25	0.5	0.75
sweatshirt	top	0.75	0.25	0.75
buttonup	top	0.25	0.75	0.25

There are also two accessories, a coat and sunglasses, used when extra warmth is required and when the UV index is high, respectively.

The premise of our system is that we will:

- Be given inherent properties of a user's articles of clothing, as well as an initial “liked” value.
- Collect external data such as temperature and UV index.
- Take input from the user on what they care about in the day’s outfit (temperature appropriate, an outfit they like, or looking fashionable).
- Search the articles of clothing by the properties specified, then apply how much the user likes the article to get the best article for each category: shoes, bottoms, tops, and conditionally, accessories.

- Take input of what the user chose to wear and consider that in future recommendations.

Not all the steps are yet complete. However, the framework set still allows us to filter by certain properties and the users opinion on the article of clothing. The statements as follows give us an outfit where the user wants bottoms that are warm and a top that is warm, as well as a warm accessory and an accessory for the high UV index.

```
<?x --> (&, [liked], shoe)>?
<?x --> (&, (&, [warm], bottom), (&, [liked], bottom))>?
<?x --> (&, (&, [warm], top), (&, [liked], top))>?
<?x --> (&, [warm], accessory)>?
<?x --> (&, [high-uv], accessory)>?
```

When run in the NARS GUI[1] and put into focus mode, we get the following output.

```
Answer <{combatboots} --> (&,[liked],shoe)>. %0.75;0.89%
Answer <{sunglasses} --> (&,[high-uv],accessory)>. %0.80;0.89%
Answer <{coat} --> (&,[warm],accessory)>. %1.00;0.98%
Answer <(&,(&,[warm],bottom),{jeans}) --> (&,(&,[liked],bottom),(&,[warm],bottom))>. %0.90;0.72%
Answer <(&,(&,[liked],top),{longsleeveshirt}) --> (&,(&,[liked],top),(&,[warm],top))>. %0.75;0.60%
```

This demonstrates that NARS chose combat boots, jeans, a long sleeve shirt, a coat, and sunglasses for a very cold day with high UV.

Alternatively, when asked:

```
<?x --> (&, [fashionable], shoe)>?
<?x --> (&, [fashionable], bottom)>?
<?x --> (&, [fashionable], top)>?
<?x --> (&, [fashionable], accessory)>?
```

NARS produces:

```
Answer <{heels} --> (&,[fashionable],shoe)>. %0.90;0.89%
```

[1] Can be downloaded here: https://github.com/opennars/opennars-lab/releases/tag/v3.0.4

Answer <{slacks} --> (&,[fashionable],bottom)>. %0.75;0.89%

Answer <{sunglasses} --> (&,[fashionable],accessory)>. %0.60;0.89%

Answer <{longsleeveshirt} --> (&,[fashionable],top)>. %0.50;0.89%

Answer <{buttonup} --> (&,[fashionable],top)>. %0.75;0.89%

In this case, when asked to produce an outfit that is fashionable and comfort is disregarded, we get an outfit consisting of heels, slacks, a button up or long sleeve shirt, and sunglasses. When two articles both have similar values for desired traits, NARS will often answer multiple times when each conclusion is reached. In this way, this system could be used to suggest multiple products that fit the user's use case, similar to Wang, 2003, as well as preventing the overfitting issues seen often in social media algorithms where no new content is recommended, preventing the system from learning the users new preferences.

Future Work

There is much more work to be done before NARS can be used as a fully autonomous recommendation system in this context.

- External data is not yet considered in the recommendation. To do this, we need a way of defining these properties on a scale and taking the scaled data into consideration in our prediction. For example, how warm is warm enough to not wear long sleeves? While the system aims to learn these behaviors for each user, an initial value will still need to be provided for initial judgement and to avoid a similar issue as a traditional recommendation system's "cold start" issue.
- Input from the user on their personal preference for the day is not yet taken. To do this, we need to design a GUI or some other way of the user inputting parameters. Currently, we just ask the system questions manually tailored to the day.
- The user's actual choice of outfit and its implications are not yet considered. To take this

input, we would again need some type of input from the user.

- Currently, the system abstracts articles into single types. More research must be done into abstracting individual articles, such as “shortsleeveshirt1” having more specific colors and properties than its inherited “shortsleeve” type. Color theory would be an interesting way to allow the system to decide between similar articles.
- A direct comparison in performance, efficiency, and scalability would allow evaluation of the proposed system as opposed to more traditional approaches.

Conclusion

Overall, it is clear that NARS has high potential in the realm of recommendation systems. While the system needs a lot of work before being adopted in any high-stakes production environments, the premise bears promise in mitigating some of the issues with traditional recommendation systems, particularly data sparsity, and the tendency of recommendation systems to overfit and stop recommending any new content. Further research will be required to analyze the specific advantages and potential disadvantages of this new approach as outlined in the future work section.

Works Cited

Kalideen, M., & Yağlı, C. (2025). Machine Learning-based Recommendation Systems: Issues, Challenges, and Solutions. *Journal of Information and Communication Technology*, *02*(Special Issue), 06-12. https://www.researchgate.net/publication/388959637_Machine_Learning-based_Recommendation_Systems_Issues_Challenges_and_Solutions

Meta. (2024). *2024 SUSTAINABILITY REPORT*. Meta. https://sustainability.atmeta.com/wp-content/uploads/2024/08/Meta-2024-Sustainability-Report.pdf

Roy, D., & Dutta, M. (2022). A systematic review and research perspective on recommender systems. *Journal of Big Data*, *9*(1). https://doi.org/10.1186/s40537-022-00592-5

Wang, P. (2003). *Recommendation Based on Personal Preference*. https://cis.temple.edu/~pwang/Publication/preference.pdf

Wang, P. (2011). *Rigid Flexibility*. Springer.

Wang, P., & Awan, S. (2011). *Reasoning in Non-Axiomatic Logic: A Case Study in Medical Diagnosis*. https://cis.temple.edu/~pwang/Publication/Diagnosis.pdf

Wang, P. (2025). *Non-Axiomatic Logic - A Model of Intelligent Reasoning (Second Edition)* [Unpublished manuscript]. Department of Computer and Information Sciences, Temple University.