Focus Group - Traditional Methods

Questions

Q1. Have you ever developed an RS of any type?

Q2. Do you think the development of recommender systems could be automated or automatically generated? If so, which parts, specifically?

Q3. What challenges have you faced while developing recommender systems using traditional methods?

Q4. Do you think that the time spent is cost-effective? If I asked you to switch to another domain, how much work do you think you would need to redo repetitively?

Q5. What are some common mistakes that may occur while developing recommender systems using traditional methods?

Q6. In your opinion, what are some potential benefits of using traditional methods, and what are some potential drawbacks?

Answers

Participant 1

Q1. No experience beforehand.

Q2. I agree with you, but it very much depends on the type of data. But I also think that it's always good if you have the choice, for example, for particular things. For example, depending on the context in the case of books and the case of movies, in the case of theater or the other example that you gave for the tracking, I always remember, for example, in Brazil, we have problems related to violence. I would like to take a route that is peaceful, or there's no shooting, because sometimes I've been in trouble there because of this.

Q3. Maybe only to, because maybe there are things that are, I'm forgetting the words, sorry. But the things that are shared for all the domains, I think that this is an important thing that you could use for every domain. And because of this, we will simplify. But of course, there are the things that we commented that are specific.

Q4. If you have a good, let's say, base that could be used in any domain, I don't know if this is possible, but as more you have this, of course, it will be easier to adapt to the things that are different, that are related to that context. But this is challenging.

Q5. For example, the thing that you commented, you have only two options, which is not good. The things that you have to... As you told, you have to start again from scratch, because this works here, but it does not work in the domain. So it's just generalization, let's say.

Q6. I would say explainability.

Participant 2

Q1. It was for recommending events for our e-commerce selling tickets for events, like concerts, sports, theater, these things. Actually, it was not something that was needed for an industrial application. It was just not for playing but for just looking at how it would be possible to recommend this kind of things, and I used a mixed approach between collaborative filtering and content-based. The collaborative filtering was exactly as you showed us, and the content-based was based on the similarity among genres, authors, kind of event, but I didn't test the accuracy and like recall and things like this. It was just trying to implement. I didn't check the approach functionality.

Q2. But to answer the question, I would say that the part that after you have all the vectors associated to maybe users or scores or like for describing the items that you are recommending. Once you have those things, you could use some part of automated generation. And actually there are libraries that do this. You use sklearn and they add something. But for the part they are saying, like, you have to extract all this metadata, all the vectors that you need. Maybe you can try something of automated generation, but you need a model that specifies that you need to provide the tool that generates the requirements in the recommender system some information because they can be different, as you were saying, time by time according to the domain, to the items that you recommend. If you are using a library for computing the similarities or the scores or whatever, the part that you should try to generate is the part that binds your domain that is different time by time with the library that you use. So I think that is the part.

Q3. I would say representing the domain, I would say, because you could follow a pattern, but you should be able to translate that pattern, so to adapt that pattern to your specific case, and also the challenge is always to get the data that you have, transform it into something that then you can actually use, because the recommender systems at the end they use some mathematical functions, so you should be able to embed all the information that you have and translate them in a way. I think that they all use vectors, so you have to be able to embed all the information that you have into the vectors, so you have to set which are the properties that you have to represent as boolean values or as categories, so a discrete set of elements, or maybe you have some continuous variables. So I think that's the difficult part because then when you are selecting or choosing similarities, if you have a set, so categories for a given property that you are considering, one and three are as different as two and three, so you should be able to represent that thing the right way, otherwise you may introduce biases.

Q4. I think that there is a lot of work to be done again. So I was thinking to the example that you showed us at the beginning, so if you have movies, maybe you are not using the title for recommending because you have a lot of genres that you can have, you can use collaborative filtering also. But like if you are recommending books, maybe what you can do is use a different, maybe books are similar to the books, but anyway, let's imagine that you should compute as you did the word embeddings for the books. And then you use a vector where you have the document, the term frequency, the inverse term frequency for each book. So what if a new book appears that has a term that is not in the other one? So you have to extend the vectors of all the other books, and I think that's a bunch of work, even if you are in the same domain at the same point. Yeah, like we do work even in the same domain. So if it appears a book that has in the title or in the, how to say, the short description, I don't know, "the elephant," I don't know, then how you should update all the other, I don't know, I think that there's iteration. Yeah, I think that you have to do another iteration, how you can be sure of the evolving, so it's, yeah, I think that there's a bit of work, just like maintenance of your own slides, something like that.

Q5. The one that Participant 3 said, the problem of the granularity, this can be a mistake.

Q6. I would say explainability, but I think that that's a benefit because it's easy. Maybe that you can rely on well-established patterns, so you know what to expect, because you know exactly what you're doing, and then you have the control over the data, but it's an effort.

Participant 3

Q1. No, I actually used this movie data source before. It was for a course during my PhD. But I used this same dataset that looks similar at least. The MovieLens one? Yeah. And the goal was to also find similarity between users based on that, the likes in the movies, but we were using machine learning to test. However, the goal was not much focused on the recommendation itself, but to improve the machine learning performance. So we calculated the recall and usage.

Q2. I mean you mentioned a lot the dependency on metadata, so they have to be the first step, right, into automation at least. These scenarios like music, the park, and the movies, they also have this category, and trying to abstract from that, so you can apply the same approach to any domain that is this simple that is only classified by a category. So yeah, I think that's more or less clear from your presentation that that's a fixed point that can be used. I agree with you, but of course it very much depends on the metadata that you have. I just want to comment on something that Participant 1 mentioned, the safety part. Is this something that you also understand as a pattern across domains? Because I was trying to like, how would safety look in a movie recommender for instance? Would it be like the age of the user, there'd be a parent providing something like that that should be considered? Of course, yeah, you're right. So that's a different layer of only considering like automating the categories. Okay, all of them have categories and that's easy to automate, maybe, or online maybe is the first situation. But how do you automate the safety part, constraints basically? Yeah, yeah, that's a nice point. So like in our opinion it's more, for example, the automation should, it's more like on the, not more on the recommendation side but more like in the definition of the domain side, let's say, how to define the domain, or more, that's the challenge. Yeah, is that, or like for example we have like, over the future content-based we have like, could it be automated as well, or just the domain side?

Q3. Yeah, I think, yeah, also the granularity then you convert like these categories. So it might affect the calculus in the end.

Q4. I think the complexity will come from the constraints, right? What we were talking before, the safety part of applying the algorithm, because if you change categories it is a bit easier to go from one domain to another, just the metadata or the column names that change when you have the array. But then the extra step of understanding, but what does this domain have of different besides the categories, that is where the tricky part I think starts. Okay, so like anything besides that, let's say, you can reuse it from other ones as you said.

Q5. And also I'm going to go back to the safety point, you don't have a way to insert constraints. I mean, you will have this recommender, but then what you build on top of it is like the guardrails in the element. It can generate a lot of bad things, but then you have guardrails on the way to filter out.

Q6. No, I mean, I can be a bit biased because this is a typical trade-off, right? Either you go domain-specific and you can be as customized as you want, but that comes with the cost, as you can see. And also, it got me thinking a bit on when you want to add a new category, maybe if you have an automated tool to create the categories, it's easier to transform it to a more granular one, that's it. Okay. Today we're going to have insights, additions, questions, comments.

Participant 4

Q1. Yes, I have developed at least three recommender systems for different application domains. So I mean of course, I worked in Italy, Bari. So I worked with a recommender system for music, about songs and so on. And we developed also collaborative filtering and content-based recommender systems for music. And then when I came here to L'Aquila, I developed another two recommender systems, but for software engineering. So maybe you can consider it as four systems in total.

Q2. I think so. I do think that the development of recommender systems could be automated. For, so as you can see with some libraries you can, let's say, just get some lines of code and then you get a collaborative filtering recommender system, right. So those parts for working with matrices to handle matrices, it's very simple and you can do it very easily, right. And also I think it's like now you don't have just only two dimensions, you can have three when you consider one more dimension for something like with the recommender system for tourism, you can consider not only the user and item but also things like weather or the date, the day and so on. So everything could be easily created and now it's not really a problem anymore, right. And apart from that I think the part for evaluating a recommender system could be automatically done for, no issue, right? So you generate some recommendations and then you compare with the ground truth data and it can be done very easily. And for now with generative AI we could even do more, right. So you could develop a recommender system but just by asking AI to generate for you, so that's even more convenient. So I say yeah, I think yes.

Q3. So of course there are some challenges when we develop a recommender system using traditional methods. So let's say when you need to handle a lot of data, like you have a big matrix with rows and columns. So that could be of course with several thousand of rows or several thousand of columns, that might not be an issue, but just think about the scenario in which you incorporate more data, then that could be an issue. So you need to consider the way you deal with very huge matrices using matrix factorization for example, okay. And in that case carefully, let's think about the case of YouTube, right, you have a very huge matrix of items and users, so in that case that could be very challenging to cope with this amount of data. And okay, of course, so the question is the challenges when just developing, right, not for, okay, so let me check. Yeah, right. So when you get more data, a traditional model like collaborative filtering, you have to run again. Whenever you want to get a recommendation, you need to run the whole process again. And when you increase the amount of data, it may take more time. Yeah. So this is the challenge we can see, I can see from my side. And when you deploy, of course, the lack of data, the miss, you don't have enough data, let's say, to train the system, and in the end you may not get very good results. So the same problem for any machine learning system.

Q4. So I think yeah, the matter of representation of data. So if you have already developed a system for doing something and then you want to switch to another domain, so it may take some time to find a way to represent the data, but then most of the work could be reused for a new domain. So I'm saying this because I had a cross experience because I switched from music to software engineering. So during, you have to find a proper way to collect the data in the first place and then try to find to represent the data in the right format, and then you can reuse the system you developed before for that domain. So I think it's not a big problem. Sure.

Q5. So this is a good question, common mistakes while developing recommender systems using traditional methods. So I mean yeah, depending on the data you have, right. So we said from before we have collaborative filtering and content-based. So depending on the availability of data you may try to find a proper system for that type of data. So when there is the problem of course, let's then collaborative filtering may not be very effective. So you can think of using the other type of recommender system. Choosing the right technique is the thing that we need to think carefully before starting to design something. So the mistake could be you use a collaborative filtering when you don't have enough background data. And hybrid could be a very good choice. So depending on the availability of data you can go for this or you can combine both.

Q6. So using traditional methods could be useful when you have a very, let's say, generic type of recommendation and you can convert the raw data so that you reuse the traditional methods for raw data. But then yeah, drawbacks maybe there are some other, let's say, methods that are even more suitable but you are not aware about them, so you just follow the traditional path and you just don't care about the other. So keep an eye on the other techniques, the other methods that could be more relevant. So that is why we think, I think, that it's very important when you, let's say, start developing a system, you need to think about your data and also start investigating the related work, so literature review to see what people have done so far on this. So we need to think out of the box, not just be focused on some very specific methods, and then we may be very limited in the sense that we could not have a very wide outlook of everything.

Participant 5

Q1. No, I haven't developed any recommender system.

Q2. Yes, I think some parts could definitely be automated. With an automated approach, the design, fine-tuning steps, and deployment can be performed automatically. In this way, you can focus on what the system should do at a higher level. This could also make it easier to try different algorithms or pipelines without rewriting everything manually.

Q3. Traditional methods involve a lot of manual coding, which takes time and can easily lead to mistakes. Moreover, if you want to switch algorithms or adapt the system to a new context, you often need to refactor or rewrite large parts of the code.

Q4. I don't think it's always cost-effective. When switching to another domain, you often have to redo many steps from scratch, such as data preprocessing, evaluation pipelines, and even parts of the logic. Moreover, you tend to remain tied to a familiar tech stack. Even if another domain would benefit from a different tech stack, switching becomes difficult because so much of the existing code depends on specific tools or libraries. Working at a higher abstraction level reduces this problem because the model stays the same, and you just perform automated steps rather than rewriting code.

Q5. A common issue is making the system harder to change. For example, developers sometimes fix algorithm choices too early, mix the recommendation logic with low-level technical code, or build a pipeline that isn't modular. These things make the system less flexible and harder to maintain. Another common issue is tech-stack lock-in. When everything is tied to specific frameworks or libraries, it becomes difficult to move to another stack that might actually be better for a different domain. This risk is much lower when you work at a higher level of abstraction, because the logic is defined independently of the underlying platform.

Q6. Benefits:

- You get full control over the implementation, which makes debugging and fine-tuning easier.
- It's highly flexible and great for experimenting with customized algorithms.
- It integrates easily with existing systems, since you can adapt the code to legacy systems, internal APIs, or custom workflows without constraints.

Drawbacks:

- It takes more time and requires strong programming skills.
- It's easy to end up doing repetitive work when moving across domains or algorithms.
- You can easily get locked into a specific tech stack, which limits long-term adaptability.

Focus Group - MDE Approach

Questions

Q1. Was it easy to understand and use the approach?

Q2. Did the approach reduce the effort of developing a recommender system?

Q3. Do you believe the approach reduces error-proneness and time-to-market (TTM) through automation compared to manual development of the recommender system?

Q4. How well did the approach allow you to experiment with different recommendation strategies without modifying the code?

Q5. How effective is the approach in handling diverse domains?

Q6. Would you consider using this approach in future projects involving recommender systems?

Answers

Participant 1

Q1. I think from the MDE part, yes. I mean, as a person that usually develops things with MDE, it was a straightforward process, kind of algorithm or something. If it's content-specific based on the context that you want to base the system on.

Q2. At the end of the day, if you put a high-level abstraction in the same domain, like as you say in the cities or whatever, and you classify all the elements that you are interested in in a city and you put it in a metamodel, and then you derive the recommender system, do this like 1,000 times, do this easily to solve, yeah.

Q3. The error-prone part is one of the bases of model-driven engineering. So, using MDE when you, I would like to say, automatically reduce error-proneness from using this class and the relations that are already established. So until you run the model from the main metamodel that you already tested, yes. I think it reduces from that side, from the side of the algorithm to generate the score, to blah, blah, blah. I think the main point of interest of this paper. Yes, for the error-prone, yes, yes. Maybe if I can say something that is not 100% like yes I agree, time to market maybe the first time that you have to develop it, if you are not 100% a model-driven person, maybe it does not reduce time to market 100%, but if you think as a long run, as for example that I have to develop multiple recommender systems or I want to update my recommender system or blah blah, yes, at this point, yes. Maybe it's the first time that you use that and you have to apply that. I'm not a humanities person, but I can imagine that maybe a person is more friendly or more known with like, okay, I just put some lines of code to develop my recommender system, but if I actually think about, as I said before, updating it or I don't know, changing something, maybe in this case, the approach is more useful to reduce time to market.

Q4. As the fact that it is model-driven and separated from the model that you develop, and you just, if I don't remember wrong, select another model or something like that, another strategy or whatever, it's easy also to experiment, no? See what happens with the same context model using different strategies. I think this is a step forward. It's easy to experiment with that.

Q5. I don't know, with more complex domains, if the general metamodel that you say is enough and represents all the possible concepts, let's say. In the case of, you made a good example, in the case of the movies, if you have to manage the score, I think the answer is that it is possible to reduce that.

Q6. But as, yeah, I suggest him just to, for example, there is the IMDB to see what happens. For example, I don't know, from these databases, this film scores like 'three,' from this other database, it scores 'two,' but from a specific user, let's say, preference, at the end, it's not two, it's not three, it's like one. Okay, so maybe it would be interesting also this degree of knowledge. So, if it's possible to easily increase the knowledge of the recommender system and to rather play easily with this, yes, I would consider it.

Participant 2

Q1. Yeah, but with understand you mean to understand how to use it, how to use? Yeah, but what I understood is that part that Arianna told describing is what you told, or I'm wrong, I don't know, that you want to improve this graphical... We also discussed, you use API in a way you have it behind, you were saying in a way beyond this could be useful also for this yes. But to me, just to go to the question, it was easy to understand because I had this background in MDE as I said. So the point of view to me is easy to use and understand.

Q2. Yeah, to me, what I remember is a matter of developing a new model for the domain that we have and connecting with the new weaving models, right? Yeah. So, to me, the developer, all the parts related to the recommendation. Starting from this, starting from the fact that you have to be an expert in MDE, I can answer that it's easy.

Q3. I agree with that. Of course, for the error-prone, and time to market is—I want to say yes, in the sense that, okay, last time we had some errors, but I understood the idea. If we take into account the fact that the complete work, complete without errors, of the framework, it seems to be a good framework in order to have this time to market, um, yeah, because I imagine that I only have to build the metamodel of the domain and connect with the recommender system model and then you have all the graphical part. Here I saw Picto just as an example, probably if you have Picto you would like to free the APIs and it seems that it's a good framework.

Q4. Agree with Participant 1.

Q5. I can answer in this way. What makes this effective? The MDE part. Because of, yeah, to me, if I only have to create a new metamodel for the domain, I think for me this system is effective. I always imagine the project that I had in the past. Okay, as I have already said, it's a mess to build the domain, but once we have the domain, it seems with this approach that you can connect a recommender system for the domain easily. So to me, thanks to the MDE part, it's highly effective.

Q6. Yes, but you have to convince me that your framework is working. It's very promising. I'm curious to use it, because I use this stuff every day, you see. And if I can play with this stuff, it would be very interesting.

Participant 3

Q1. I mean, from my point of view, that I am not in MDE, I do know like the basic concepts and I've used Eclipse. I did find, and maybe this is because of my background, I did find it a bit hard to, I mean, literally look at the screen and understand what's happening in terms of like the items and how do they, yeah, it's more of how do they match to each other, right. But then I think this is just because I'm not used to the, you know, to that IDE and how things work there. Yeah, I think it's more of the technology. But from the point of view of, I mean, I've programmed many things and I do think that programming a recommender system that is generic seems very hard. And from the practical session, I really had the impression that you made that thing easy to do. I did have this feeling. I'd say that from this perspective, I'd say it's easy to understand. I feel more comfortable to implement using your tooling, despite the technology.

Q2. I mean, I... It feels like it reduces the effort because it abstracts from... Also remember this part which seems very interesting that you're selecting if you can automatically select different engines to develop. I think this by itself reduces some effort in the development. So, yeah, I do think so, yeah.

Q3. So I do have two things about error-proneness. And what makes me... Yeah, I'll explain them. So the first of them is that you have to assume that the models that you're generating are correct by design, right? So if you assume that they are correct by design, that your specification is formally verified and all of that, then I do agree that the approach reduces error-proneness, especially when it comes to writing the logic of the code that you have to do. And one interesting thing that I think about error-proneness is that the errors will be more on the understanding of the domain, right? To the detail, yeah, so because you're specifying just the domain, you do the separation between writing the code and the logic that will be used and specified. The only kinds of errors that would appear are the ones in the domain that would also appear if you're coding in any other way. So thinking about that, it seems for me that indeed, the approach would reduce error-proneness. With respect to time to market, I do like the argument that on the first time you develop, maybe not because, well, you're developing it from scratch anyway, but once you have to, I don't know, migrate it to another domain or evolve your code base, then indeed, I think it reduces time to market.

Q4. Agree with Participant 1.

Q5. I don't know about effectiveness. I felt like it improves in efficiency, definitely. I think it makes the process faster. On effectiveness, I don't know because I don't think I understand what effectiveness could be here. Yeah, yeah. So, only to complement what I said before, it's because I had the impression that I do find the separation of concerns very interesting. Because you don't have to worry about the logic of how things are implemented, but you have to worry about the logic of the domain. And then you have to worry about that anyway, with your approach or with the other approach, right? So I think it frees you from the logic of how do you write the code and then you can focus more effort in understanding the domain, which is something really cool.

Q6. Yeah, I mean, I think this would be very interesting to see. I asked in the past meeting, but whether you could recommend software components for, I don't know, I'm interested in self-adaptive systems and in self-adaptive systems, you need to change the behavior of the system during the execution. And sometimes it will have like a few options, right? And then what to choose and when, right, maybe I would, yeah, I would do something like that. Yeah, it sounds cool. I mean, it sounds easy to, you know, choose which recommendation do you want to pick? And now you need to work on the domain, which then would be, I don't know, software that you want to change. I'd like to know if this would work in robotics, for instance.

Participant 4

Q1. Yes, same from my side, everything was self-contained. So I would say that was straightforward. The only drawback with my machine was that it consumes too much power. But you're lucky you're not present in the first one. Yeah, that's stressful. So thank you. Another thing I would say—maybe with the water, you know, it's pointing out because, like, the recommenders were using like scenario N1 and N2. I was expecting like having some concrete example, if it's about a movie, a concrete name of a movie, something like that, just to give a clear feeling.

Q2. Same as me. So we have reduced since we are abstracting most of the things and we focus on the details, on the logic. I would say that the effort is much reduced compared to the classical development like coding and other things. Since we have a model, then other things you can have it almost for free.

Q3. Let's go to another one. Because I will say the same thing. I'm going to say for sure, yes.

Q4. Agree with Participant 1.

Q5. Yes, I saw that your main metamodel is generic, which can also accommodate different other use cases of different domains. I would say that this approach is effective. Also using the weaving models, doing this kind of relation.

Q6. The approach is really nice. Last week, I was searching to find some metamodels related to the use case that I want to work on. I had to search online, and it's very hard to find what you want. So having this recommender system, which can easily recommend you a metamodel or a model you want.

Participant 5

Q1. I will say for me it's very easy because they have different things but yeah the essence is quite similar to the recommender framework I developed during my PhD, so for me it's easy to understand. I will say to use, because connecting it's not difficult to listen from the traditional way of developing recommender systems.

Q2. Yes. Yes. Because developing a recommender system from scratch is very tough.

Q3. Yes, in my opinion it would reduce the error-proneness and TTM.

Q4. I think that in the dashboard you could also select the algorithm, so I would say easy, like, well, since you have just to change that instead of recoding everything.

Q5. Highly effective, because you only change the domain model. So time will be only developing that domain model I guess.

Q6. Yes. The recommender system. Yeah. Because mine is MDE for recommender systems and recommender systems for MDE, not for traditional recommender systems. So for traditional recommender systems, yeah, it will be interesting. And I really like this project you had in the park, those types of things, it's interesting based on the interest, particular domain interest a person will have.