Master's thesis : generating gaml code

Nicolas Renout

September 25, 2025

Contents

1 Acknowledgements

I thank the ACROSS lab for the warm welcome, especially Alexis Drogoul who has supervised my internship. Many thanks to Ada Diaconescu who has supervised my internship on Télécom Paris' side, and who gave me the lead to pursue the internship in Vietnam. Many thanks to the laboratory for allowing me to prepare PhD interviews on the work hours, it has been a huge boon to me.

2 Short summary

It was my first time travelling in Asia so the knowledge acquired during this internship has been both technical and cultural.

In order to realize all of these projects, the Across developed a platform named GAMA, which is specialized in agent based modelling simulations, using the gaml language. As of now, this language is seldom used outside of the lab, resulting in a lack of performance from LLMs like Claude or ChatGPT to generate working gaml code. Because so little gaml code has ever been written, fine-tuning revealed itself to be impossible, therefore another method had to be used to retrieve knowledge, this method being a graph RAG of the documentation.

It is important to note that while graph RAG is a well documented technique [2], the process of "giving" knowledge to the LLM using a graph rag has, to my current knowledge, never been done before. A good part of my internship went into creating the methods to use such a graph and to constitute it, as well as constraining the answers in order to get the best code as possible at the end.

At the current date in which I am writing this report, my subject cannot be considered as complete. Much work has been done, and while I can say that creative work has been done and some interesting results have been obtained, I did not manage to create a LLM that is able to produce gaml code using asa source of knowledge a graph of the gaml documentation. It is also of note that Gemini in now able to take millions token inputs, enough to give it a whole language documentation in each prompt, therefore this internship subject could be completed using this method.

3 Lab presentation

My internship has taken place in Hanoï at ACROSS (Advanced Computational Research On Sustainability Science) Laboratory, from IRD (Institut de Recherche pour le Développement), Thuy Loi University, Hanoi Architectural University and Toulouse Computer Science Research Institute. Across is in fact a collaboration from different actors from different countries, resulting in an interesting work ambiance. It is important to note that technical roles are mostly occupied by french people while social research is mostly done by vietnamese people, even if exceptions exist. There are other non-technical interns, such as artists : each year the laboratory is hiring one or two artists interns, almost exclusively from the school Emile Cohl at Lyon.

ACROSS is also a place in which some people come to complete a PhD related to Vietnam (such as the distribution of water in the Mékong delta), leading to very interesting conversations due to the broad scope of work done at the laboratory. It is also to note that a lot of interns are engaged in the technical part of the lab (around half of the technical people are interns, with half interns coming from Télécom Paris). I believe that ACROSS is following this strategy because having a six month intern is enough to add a piece to the work of the lab or to explore a trail before deciding whether or not this trail should be followed. Having so much interns is undoubtedly useful for exploring or realizing multiple projects in parallel, but it force the intern to be highly autonomous. As a autonomous person I absolutely loved this fact, however I can see cases in which the intern would be less receptive to this methodology.

The laboratory across is working on several projects, mostly centered around multi-agent simulation, the most notable project being Starfarm, aiming at modelling the rice farming in the Mekong delta. But another task of the laboratory is to use Virtual Reality to transmit knowledge using serious games, aiming at simulating floods or ecological gestures. What strikes at first glance is that the laboratory has a lot of different roles, not only re-

search but also political. For instance, we have been hosting a Chinese class, we have been visited by a Chinese delegation, Alexis Drogoul (the research director) had to prepare for the coming of Macron in Vietnam and so on. In the same way, the majority of Alexis' work was not to do research bu to overview the functioning of the laboratory, meeting people and present the lab's actions in order to get funds. It was really enlightening to realize that the director's administrative work had so much importance in the life of a research laboratory.

4 Introduction

In order to realize several of its projects, ACROSS laboratory had to create the gaml language, seldom used outside of the lab. This language is to be used by experts in social sciences, which is arduous at the current date because of the language structure and the totally understandable general lack of knowledge on coding from experts in sociology or biology. My internship takes place to improve the learning curve of gaml code and simplify its use by experts of other background than computer science, bu also to help the lab to work with LLMs, set up pipelines and so on.

5 My mission

5.1 Objectives of the internship

When I passed the first interview, the description of the job was that I would be studying how to create gaml code with AI, which was quite vague. Therefore, we determined when I arrived that the objective would be to create a pipeline using LLMs to create a gaml code based on a textual description, and this tool was destined to be used by researchers of social scientists who use the GAMA platform to modelize their work, but are not necessarily proefficient with code. It is important to note that this is not absolute, and such a tool could also be used by people proefficient in gaml to helm the create new models or templates.

5.2 STATE OF THE ART - A REFAIRE

5.3 Producing gaml code

5.3.1 Fine Tuning

I firstly wanted to fine-tune a LLM in order to assess if it was usable. For this, I essentially recovered all of the code gaml available on github and huggingface since during a precedent internship a dataset of gaml code has been made, some numbers about this code is available at figure 1. As this figure shows, the number of available code is quite low for a fine tuning (around 600 models, each relatively short), and the diversity of the code is lacking (since a single person is responsible for 10% of the whole dataset).

It is also important to note that I did not fine tune the LLM on gaml code, but on answering a prompt with gaml code, which meant that for each model that I retrieved, I had to find a way to augment my database to have prompts corresponding to the model's functionality. Therefore, I used Claude to look at each code at determine a prompt that could have resulted in said gaml code. This step worked quite well since gaml is close to java, so

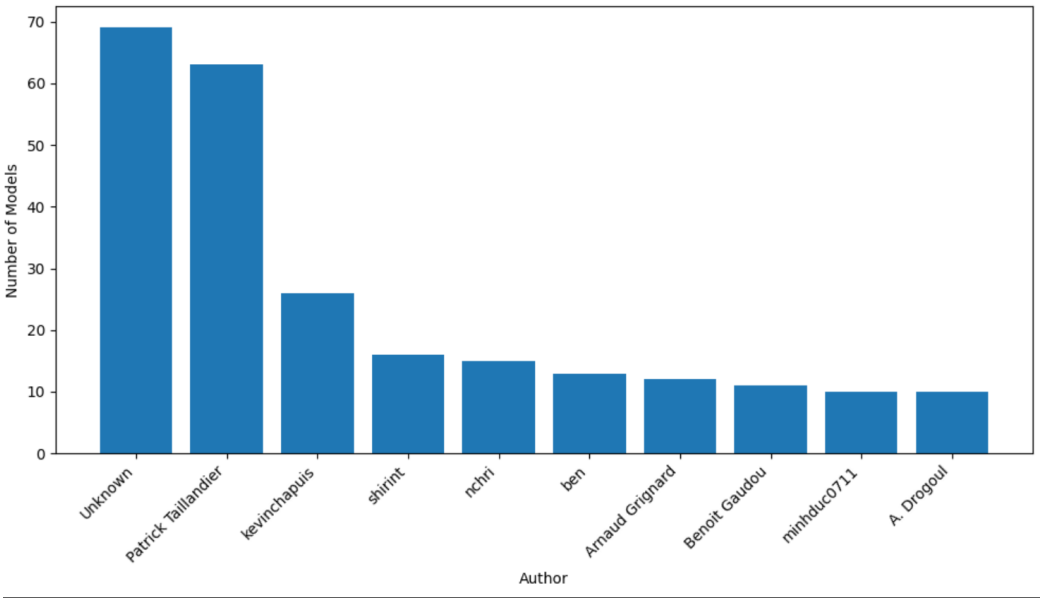

Figure 1: Number of models available on github for the major authors

the LLM managed to see some patterns and was very good at determining a prompt for the code snippet.

I then tried to fine tune qwen and I got the results figure 2, which are not inherently bad for a fine tuning since the loss goes from 2.4 to 1.4, however in our case it was still inconclusive, especially since the loss is computed using the perplexity (so how well the model predict the next word in a sequence), which is not a metrics totally adapted to code generation since writing code is not only about predicting the next word, but also about having a global understanding of the code structure. By testing the finetuned model by hand on some prompts, I saw that in fact the code outputted by the finetuned LLM was more "illogical" than the base model, can be explained because the fine tuning allowed the LLM to learn some gaml patterns, but not enough to have a good understanding of the language, hence the improved likeliness to make structural mistakes. I also tried to fine tune other models such as codeqwen, which led to similar results.

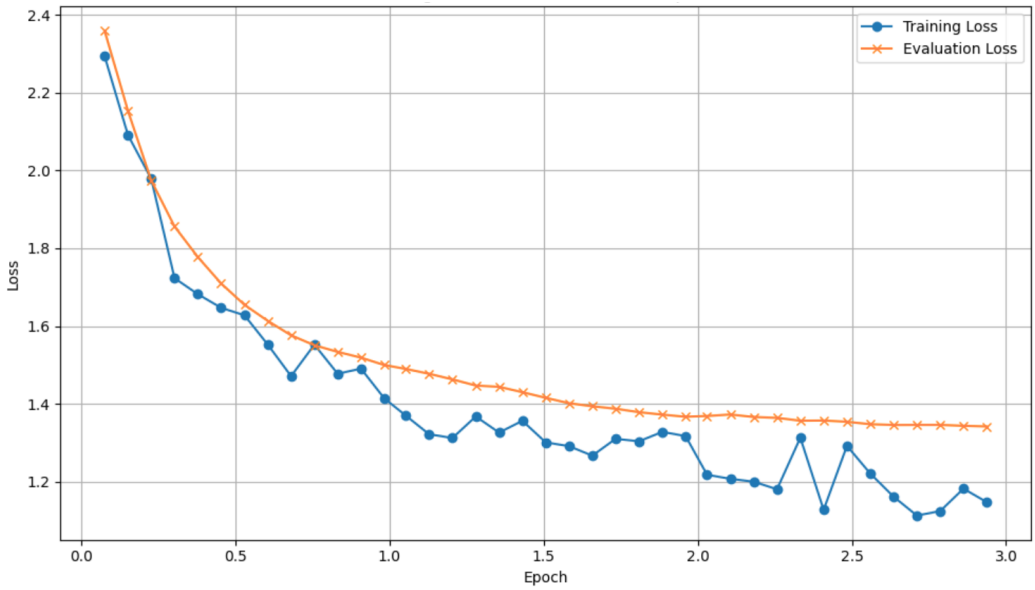

Figure 2: Loss of a fine tuning in function of the epochs

5.3.2 Methodology

As I explained earlier, fine-tuning is impossible in our case. Therefore, I firstly gather as many ideas as possible and I decided to implement them one by one to see the promising leads and the reasons of failure in other. My first idea was to have a LLM specialized in code correction going after a RAG to automatically correct certain errors and insure a good code structure, possibly with the feedback of gama. It turned out that doing this without a gama feedback did not really had a sense since it did not cost anything to put the information or correct code structure and so on in the first LLM instead of a second one dedicated to correction, therefore I focused my efforts on working with a feedback from GAMA. It turned out that the function used to get the error log from a gama server was indicating an error type, but not the position of this error type, which made debugging very arduous, especially using a LLM. I also wanted to consecrate a moment of my time to set up a fine tuning to see if the LLM could learn some gaml rules. My last

idea was to use knowledge graph in order to be able to retrieve very small and relevant parts of the documentation.

I then implemented each of these solution and after reaping the low hanging fruits I decided to focus my efforts on the graph rag, as it help de most promising results. I also had in mind a solution in which the knowledge graph was not constructed based on the documentation but rather on the gaml code, however the solution would have taken a very long time to implement so I made the choice to not try this.

In general, the methodology I use for advancing on an idea is as follow. Firstly I would decide which part I would like to improve, then I proceeded to search what solutions were available, ideally by reading skimming through scientific papers. When I had time, I took it to read them whole to get a better overview of the discussed topic (even if it was quite far away from what I was doing, since sometimes it can result in a good inspiration). Then I would design a pipeline usually with a canvas, since it is easy to have a general overview of it, and I also had to present my work at regular intervals so a schema has to be done either way. ACHTUNG INSERER SCHEMA !!!!!!!!!!!!!!! Lastly, I had to implement the solution that I designed, which took a variable amount of time. At first I wanted to test a lot of simple different things, but quickly the whole process complexified and therefore the time I spent implementing went up, especially since testing new complex things meant using more complex tools, and I did spent a lot of time setting ups new tools in the server.

5.3.3 Special conditions

Only few gaml code have ever been written (figure 1), leading to the impossibility to fine-tune a LLM on gaml code. I was able to fine-tune a model on the little gaml code I had at my disposition, however the results were far from satisfying, therefore my first task has been to find a way to circumvent this

problem. I first thought about rag, therefore I chunked the documentation of gaml in several parts and I used those chunks to do a rag.

Another constraint is that the LLMs used had to be run locally, no API keys were allowed because of the cost. The majority of the LLMs used are either in ollama or pulled with HuggingFace, the sole exception being the llamacpp docker user for streaming constrained generation.

Finally, the AI interns had a dedicated server to work with, with around 1TB of space and 2 GPU, in theory we has to regulate each other for resource sharing, in practice we never needed to compete for resources (except for storage space), I was the intern that was using most of the resources available since the other ones did not require much computationally-wise.

5.3.4 Classic RAG

In essence, a RAG (Retrieval Augmented Generation) is a technique in which we give the LLM a questions and all of the documents needed to answer this question. The LLM is doing a work of interpretation and summarization but does not "learn" anything, in the sense that its weights are not modified in any way. On the plus side, it means that changing of model is fairly easy, and all you need is a documentation and a LLM proefficient in processing language, summarize and interpret text. However the LLM need to have these informations provided each time for it to work, and since we are limited in how much token are possible to pass in each input this is a huge limit, while fine-tuning a LLM (ie changing some of its weights), or training it on a specific dataset do change its weights and make it stock more information.

The retrieval is made based on a comparison between the embedding of the query and the different embeddings of the pieces that we can retrieve. Therefore, the first step was to create our pieces that the LLM was allowed to retrieve, these pieces are called "chunks". Several parameters have been considered, notably the chunks overlap (since we wanted the cuts into the

code to have as little impact as possible) or special conditions on chunking in order to chunk respecting the pattern of the documentation (code and then explanation of this code).

While this yield results, the code had a lot of syntactic errors and confusion between different keywords. I also had a lot of confusion with other languages such as java or python. This could come from the fact that the rag is based on the question in itself, not the process to solve the question, therefore an amelioration of the code has been to use a LLM to create a logical plan (ie : create a grid, then add sheeps, then add wolves, then add reproduction mechanics,...), however the results were still unbalanced. While looking at the chunks that were used, I realized that the useful informations were drowned inside the chunks, if present at all.

This led me to believe that in order to make RAG useful, I would have to use very precise an well-connected informations, and using a graph RAG seemed to be adapted for the situation. In essence, a graph RAG has exactly the same principle as the classic RAG, however the informations are stocked in a graph instead of chunks. For instance, the information "Pierre is the brother of Anna and is born in 2018" can be stocked like "2018¡-born.in-Pierre-brother.of-¿Anna" ACHTUNG INSERER GRAPHE ET REFAIRE LE PITI SCHEMA !!!!!!!!!!!!!!!!!!!!!!!!!, which is a gain of space because only the essence of the information is stocked, not the form, and also a gain of logic since a link "brother.of" is a statement more reliable than a text describing a relation between two persons.

5.3.5 Concerning the database, RESECTIONNER ET FAIRE LALGO AU PROPRE

The database has been created using neo4j, by using the gaml documentation. I had to sped quite a lot of time parsing and automatizing the creation of the database since many documentation pages were created with a different convention, so custom regex had to be crafted for almost each category. Once

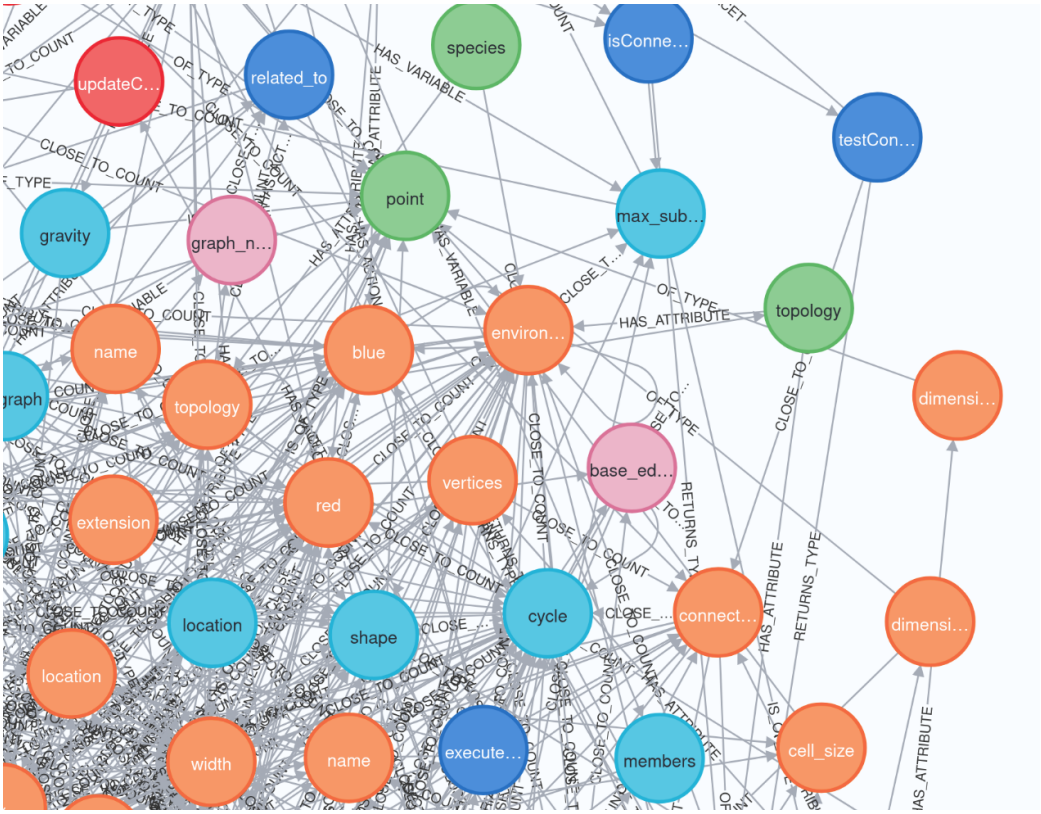

Figure 3: A sample of the graph used for the RAG

this is done, I concatenated all the information on the node and I embedded this content for a base for my graph RAG.

I firstly linked the different nodes based on the link of the wiki and some text such as "see also : +,-,=", enable me to draw connections in the graph. I also drew connections using the types used and so on, however the graph obtained after that was very sparse, so the graph rag was not very efficient since the graph is a lot of little parts of knowledge and the big selling point of using graph rag is that we can retrieve informations related to the question much more efficiently, so I had to find a way to augment the graph and add artificial connections. It is also of note that originally I wasn't sure whether to graph the documentation or code in itself, the latter would have needed a

parser to link the different elements of code in a graph. In the end I used a workaround of this method for the augmentation of the graph, but if I had more time I would have liked to try this method.

We note :

Γ : The graphe of the gaml documentation

$\Sigma = \{\sigma | \sigma \in Codebase\}$: The set of gaml code used to augment the graph

$\sigma \in \Sigma = (w_\sigma^n, n \in \mathbb{N})$: The different gaml codes, each a suit of words

γ_σ : The subgraph created from the code σ

$\Gamma_{extended} = \Gamma \cup \bigcup_{\sigma \in Codebase} \gamma_\sigma$: The extended graph

for $\sigma \in \Sigma$ **do**
  for i in len(σ) **do**
    Add w_σ^i to γ_σ
    Add the link $w_\sigma^{i-1} \xrightarrow[NEXT]{} w_\sigma^i$
    Add the link $w_\sigma^i \xrightarrow[OFBLOCK]{} w_\sigma^j$ with $j < i$, if w_σ^j is the start of a block containing w_σ^i
  end for
  for $w_\sigma \in \gamma_\sigma$ **do**
    if $\exists !(n \in \Gamma), n.name = w_\sigma.name$ **then**
      Add the link $w_\sigma \xrightarrow[REFERENCE]{} n$
    else if $\exists (n_k)_{k \geq 2, k \in \mathbb{N}} \in \Gamma, \forall n_k, n_k.name = w_\sigma.name$ **then**
      Add the link $w_\sigma \xrightarrow[POSSIBLEREFERENCE]{} n_i, i \in [[1, k]]$
    end if
  end for
end for

Once this has been done, we can proceed to the classification for the collisions. I separated these two tasks mostly for practical reasons : I had absolutely no idea if the classification would work the first time, so it seemed

wiser to write the possible collisions in the database before writing the results instead of just writing the results directly, which would lose information if the classification did not work.

for $\sigma \in \Sigma$ **do**
  for $w_\sigma \in \gamma_\sigma$ **do**
    if $\exists r \in w_\sigma.relations$ and r.name=POSSIBLE_REFERENCE **then**
      We retrieve all nodes $n \in \Gamma$ that are linked to w_σ
      We pass their description to three LLMs along with the 5 lines that are around the word w_σ, and we make them vote for the best candidate
      The best candidate is linked to w_σ with a relation "REFERENCE"
      All relations "POSSIBLE_REFERENCE" of the node w_σ are deleted
    end if
  end for
end for

In the end of this algorithm, no nodes of $w_\sigma \in \gamma_\sigma$ are linked to multiple nodes $n \in \Gamma$.

When I tried to augment the database, I was facing collisions in the sense that I was sometimes unable to determine which description was corresponding to a word of a program that I was using to draw connections between the nodes of my database. This could happen because a single word can have different meanings, for instance something that happened quite often in programs was something like location¡-mycell.location. The first location is an unnamed variable, while the second is a reference to an object, and since different objects have the attribute location, the documentation has several instances of different location items, represented by different nodes in the graph, all referring to different types/functions. In order to draw the good links, it is primordial to correctly identify which node correspond to which word in the program, since we are linking the general description of each

function together. Therefore, I had to find a way to differentiate, and I did that by giving the code snippet in which the collision appeared to a LLM to classify the description. At first I used deepseek, however for an unknown reason at some point it began to produce hard hallucinations (with outputs consisting in mostly good answers, up until the point where deepseek switch to a melange of chinese and english, and then the output for every questions was "GGGGGGGGGGGGGG". Which was funny but I still don't understand why this is possible since all of the questions are independant, meaning that a new instance of deepseek is called each time (in theory)). I then switched to codeqwen and I tried several parameters (notable how much code I gave the LLM to solve the question asked), and I finally had a pretty good results. In order to eliminate the uncommon (but still present) errors that I had, I decided to run the program 3 times for each collision in order to get a qorum and be able to decide in a more certain manner.

I then linked together the nodes that were appearing close to each other or in the same logical blocks in every programs, so that retrieving a certain node also retrieve the nodes that are likely to be useful with it, basing ourselves on the code that has already been written.

RELIRE DETAILLER COMMENT LES NODES SONT CREES EN PARALLELE

The augmentation of the graph requires a lot of programs and there are word with collisions in basically all programs, so we have to solve multiple collisions and each solve requires multiple calls to llm, therefore the computation time is high. To top it all, the call and updates to the neo4j database does require time, so after some tests I estimated that I needed approximately 8 days of calculus to augment the database. Since that would be a long period and at this time I did not have any other major idea that I could work on, I decided to parallelized the process of augmenting the database. In order to do so, I adapted my code so that the augmentation of the database from one gaml code was processed by one processor, and then I also had to

adapt the way the different gaml codes were stored into the graph, I had to adapt them so that when a processor was adding modifying some nodes some tag depending of the processor is added, and only nodes that have the tag of the processor can be modified or deleted by the processor. This was kind of arduous because while some libraries allow to parallelize some python code, here I also had to find a way to prevent the neo4j graph from being broken by the parallelization.

Using this method, we are able to augment the graph by adding information of logical proximity (since words that are in the same blocks are linked) and spatial proximity (since we tag in the graph "close spatially" two nodes that are in a 5 line range) from the previously written gaml code. With this, we hope that if a node is very often related to another one, this graph will show this connection.

We will talk in the next section about the graph rag in itself.

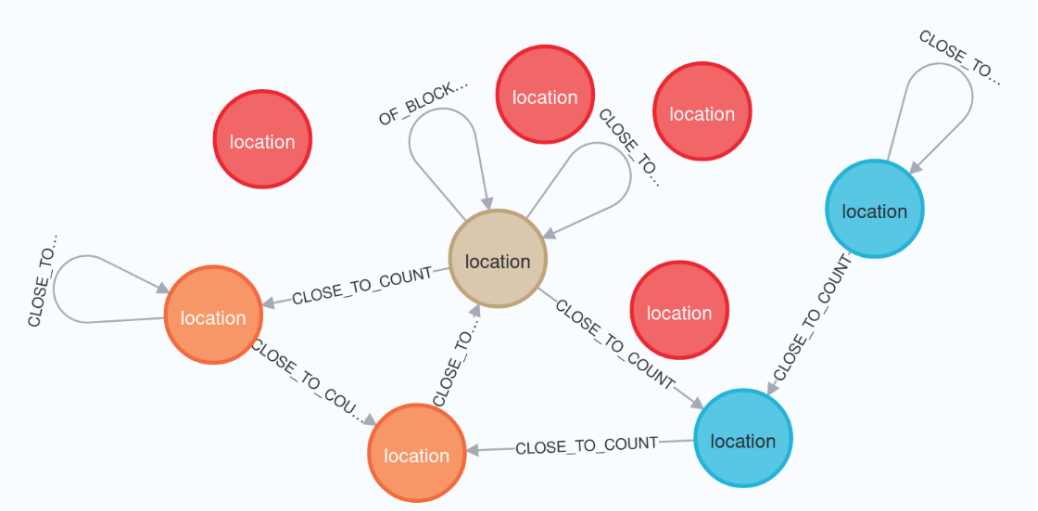

Figure 4: An example of a collision : the word location appear several times in different contexts

5.3.6 Graph RAG

The main idea of the graph RAG is to retrieve parts of the information relevant to the question asked to the author, and then to use the graph to retrieve information that is linked to the parts of the documentation that have been recovered previously. In practice, the first relevant nodes are retrieved using an embedding comparison of the question and the different nodes : all of the nodes have a propriety that contains the embedding of all the other proprieties, that is the embedding that is compared to the embedding of the question.

FAUT RE-ORGANISER LA PARTIE SUR LE GRAPH RAG (COMMENT EN FAIRE CLASSIQUEMENT? POURQUOI LA SOLUTION PROPOSEE ETC)

Usually, the node retrieved from a graph rag are determined by a shortest path or simply by selecting nodes by the nature of the links between them. However in our case it is impossible since there is no way to have a graph that fulfill the topological requirements to have a graph in which these techniques are usable because depending on the way of constructing our graph, it is either too sparse of way too dense. Therefore I had to find another way to do the graph rag, and that is to have weighted links between the nodes, and then to retrieve the nodes that are the most connected to the nodes that have been retrieved by embedding comparison. The weights of the links are determined by the number of times a link has been created between two nodes, therefore if two nodes are often used together in programs, they will be linked with a high weight.

Going from the nodes retrieved in the first place, we can extend the information retrieved using the links created during the augmentation of the graph using the existing code on github, using an algorithm close so signal propagation. In essence, the nodes retrieved originally have a signal of 1, that will be propagated to their neighbors with a ponderation based on the count of the links outgoing from the node. For instance if we have A-2-¿B

and A-5-¿C (2 and 5 representing the ponderation), a signal of 1 starting in A will propagate to B with a value of 2/(5+2) and to C with a value of 5/(5+2). We then put a threshold by checking if the signal is above a certain value, if yes it propagate again using the same rules, and if not it doesn't. It is also important to note that the amount of touched nodes is quite low compared to the total nodes, meaning that the majority of the available data is seldom used by an average user of gaml that has pushed their code on github.

At first the results were not so great, mostly because the question did not really have anything to do with the nodes concerned by the question. In fact, the rag is being computed by comparing the embedding of the query with the embedding of the nodes of the graph, but is the user needs a particular function, the node containing this function is not guaranteed to be present

Therefore, the pipeline evolved to a first LLM translating the question into a pseudo-code, and then another LLM translating the pseudo-code to gaml code (with the graph RAG). This permitted to be way more efficient as the code outputted by this method still had errors but was much more logical in the sense that it was doing more or less what the user was asking and what could be logical regarding our laws or physics. For instance I asked the LLM to represent a model of a burning forest, before the pseudo-code trick the LLM was doing very weird things, like making the trees grow if they had enough food and so on, while the LLM with the pseudo-code was much more close to what we would expect (ie a model of fire propagation). It seemed that the passage by a pseudo-code added a certain logic to the whole program.

However the major downside of this method is that the longer the pseudo-code is, the less pertinent the context retrieved from the graph become. In order to address this issue, the next step I used was to split the pseudo-code in several parts, and translate each part independently from each other. A final LLM is instructed to merge the different translated parts, with the use of some examples in its prompt. This pipeline increased by a very large amount

the computation time, but yield results more precise and logical. However the major issue I was now facing was that the high number of LLMs made the presence of errors more likely, and an error in this pipeline could propagate and really mess up the final result.

I then changed the pipeline again by changing the generation from pseudo-code to java code, since gaml is close to java. While I was persuaded that this would improve greatly the performance, I think while testing that the results were slightly worse (the metrics used were not precise enough to detect significant changes between versions, mostly because no code compiled and the errors were very uninformative). This can be explained because the pseudo code (or java code in that case) is only useful for retrieving information in the graph and laying a general idea of the shape of the code. While using java as a basis give a better shape, the information retrieved from the node is less pertinent (since the nodes are in natural language for the most part). Moreover, java code leads to more confusion and add a risk of having some outputs in java.

Once the constrained generation part has been addressed (I will talk about this a bit later), I updated the pipeline again to work with incremental generation, so that instead of translating code, the pipeline know is adding instructions to a program little by little, while always respecting the overall structure of the program. In fact, a first LLM is looking at the user's query and is instructed to give a list of things to do to tackle this query with some descriptions (the descriptions are useful for recovering pertinent nodes from the graph, as the retrieval is being done by comparing the embedding of a query and the embedding of the nodes). A first LLM produce the code corresponding to the first step and pass it to the second LLM who has for instructions to complete the given code with the second instruction, and so on. Once all of the steps are generated, the whole code is being passed into a LLM with a constraint generation, who has for role to mitigate eventual errors. I am using constrained generation only at the end since it is quite slow,

and more unstable from the tests I have been conducting, in the sense that the LLM used with constrained generation is more prone to hallucinations. I even had to change a bit the grammar to forbid the last LLM to write commentaries at the end (it was using commentaries to hallucinate). I also had to automatize the generation of gaml code to create a benchmark, and I was forced to limit the output of the final LLM in order to make it stop generating tokens if it was hallucinating.

5.3.7 Constrained generation

The last step has been to add a constrained generation to the LLMs in order to force the creation of correct code. The implementation of this has taken time because of diverse problems of compatibility, that I will describe below. I firstly translated the gama syntax to lark, which was usable with ollama models. This part required many simplifications due to how lark is working : I had to start from a .xtext file used inside the GAMA platform, used to verify the grammar of the gaml code, however lark works in a way that it is very bad at solving collisions, so I had to rewrite a part of this code.

However after debugging, since the constrained generation was outputting nothing, I had to adapt my code to do streaming generation, which is impossible with ollama (to be more precise the ollama function for streaming generation is declared bu not implemented). Therefore I had to install llamacpp on a docker and rewrite the grammar in gbnf, which allowed me to see that the constrained generation was beginning correctly but due to the complexity and the many possibilities of recursions inside the grammar, the constrained generation had too much path to explore and therefore its computation time was becoming exponentially slower during the generation. Therefore, I had to simplify the grammar, notably by removing some recursive possibilities and unused parts of gaml. The outputs of the constrained generation is now mostly correct syntactic-wise but still harbor problems. In short, having a set of rules too strict is hindering the LLM's ability to

generate code in a fixed amount of time, and a freer grammar is prone to produce errors.

In the process of gathering information on constrained generation and searching for methods that could be used for my problem, I contacted the researcher Alexandre Bonlarron, who wrote a paper about constrained generation for LLM (reference [1]), and kindly answered my questions. He hasn't been working on constraining the generation of code, but he gave me advices and thoughts about promising leads (not only about constrained generation but about the whole project), notably the use of chain of thoughts and tree of thoughts, or integrating a feedback from the compiler, but also prompt engineer the LLM or tweak the constrained generation to explicitly exclude certain errors if they were recurrent. This last point was particularly useful since I was much more focused on the pipeline problems and I hadn't thought about seriously excluding some errors, and the rest of his suggestions validated my ideas, only the feedback from the compiler was very hard to apply in practice because of technical limitations (it would have needed to rewrite some part of the gama platform, point that I passed on to the developer team but it would have been simply too time-consuming for me to address this problem since its root lied far away of my domain of expertise), and I was already making designs about integrating a chain of thought in my pipeline.

It is of note that I was genuinely surprised by the depth of the answer from Alexandre Bonlarron because of the time it should have taken him. I was expecting a short answer with some references but he really involved himself in his reply (he even asked me for updated later), which allowed me to see that contacting other researchers was something that could very well work and sometimes unlock new path of exploration.

5.4 Helping the lab

5.4.1 Coding camp

During my internship, ACROSS had planned to do a "coding camp" working with a lot of different people in a very nice location with the possibility for those who want to make extra hours. It was really fun, and the work of the interns was mostly to hunt for bugs in the gama platform and report them on the github. Because of the global gathering, we also were able to talk of our subjects with several people and have constructing conversations on them, the major boon of the gathering being the presence of experts with different fields of specialization. It was my first experience of what could be describe as work with a vibe of holidays and I found it really productive, and according to the general feedback these kinds of weeks are almost necessary to do any major upgrade on GAMA because a lot of people working on it are gathering and because they can work very late (or early) if needed.

5.4.2 Documentation RAG

While I was working on rag for producing gaml code, I also realized that this technology could help ACROSS in another way, which is to create a tool able to search documentation. This has been done using the base of the graph RAG and it is mostly useful because it is very hard to search inside the gaml documentation. However, the rag is not performant in the way that it is hard to retrieve the information about a certain subject, and sometimes the important informations in the documentation are not retrieved because of the semantic difference.

It is also of note that the graph of the gaml documentation can have side interests such as spotting redundant functions or simply exploring the gaml structure (since it is more appealing than a block of documentation, at least for beginners). I have done no work at all on this side, but using graph could be the foundation of a project in GAMA.

5.4.3 Helping on other's research

Sometimes I was solicited by a PhD student in geography to beta-test and help improve a simulation of water distribution in the rice culture. While it was not linked to my internship's subject, it was nonetheless interesting to discover other facets of research in very different fields. I also had discussions with PhD student about how to conduct an interview and what parts are really interesting, for instance understanding that the dialogue with the people after an interview is as important as the interview in itself. Since we were a lot of interns, it was also usual for us to talk to each other about our projects, which could give ideas or leads. The process of explaining one's thoughts to somebody else also forces to have a clear reasoning process, which can be useful for spotting mistakes or simply think about new leads. I regularly had chats with other interns in which we each other explained thoroughly our projects, and these chats were very useful to get new ideas or have a second sight on some things that I had overlooked, and that is especially true since all of the interns had different fields of specialties.

5.4.4 The intern's seminar

Across has for habit to schedule one intern's seminar each month, in which the interns (and sometimes the employees) each talk about their respective work with a slide presentation. This habit has been taken because the laboratory often employs a lot of interns and needs to have regularly updates in order to have a good vision of what is being done. It is particularly useful if one intern has a specific problem that can be solved using somebody else's expertise in a particular area, and presenting one's work is a good oral exercise and it force the interns to sum up their work each month, which is useful to see one's advancement. However, I do think that the lab could also benefit from having the regular employees do this since the interns generally have a broader but shallower field of expertise, meaning that they could potentially give leads to work on. In the same way, since the laboratory is a very heterogeneous space (in which social science are done alongside computer science), the intern's

seminar is also an exercise of vulgarization, therefore it is impossible to enter too much into the details. It is a deliberate choice of the lab, however I think that I would have preferred to have longer presentations with a vulgarized part and a technical part, mostly because if you vulgarize too much the interest of having experts able to give their advice is kind of lost, but if you vulgarize too little then three quarter of the room is asleep.

Interestingly enough, this seminar is also a pretext to have a common meal (of pizzas and spring rolls to be precise), during which people can talk to each other about their work subject. I found it very useful to have this informal possibility of exchange after a seminar, because it is much easier to talk in this context. Overall, it does same to be a good idea as it allow the whole lab to stay updated on the interns' work.

6 Personal development

6.1 Work wise

The thing that I find the most remarkable about my internship what that I had a huge freedom : I only had a general goal and I had not only to choose which methods I would use, which experiments and tests I needed to use, but I also had to choose what center of interest my work should focus on. This has been a huge boon since I work well autonomously, and I have been able to experiment new things on my end. My work was supervised in the sense that I regularly had to present it both to the whole lab during the "intern's seminar" and also to my supervisors during one on one exchanges. These two kinds of presentations were very different exercises : during the intern's seminar I mostly had to present the general ideas of what I was doing and why, with enough simplifications that my work was understandable by people in other fields than computer science. The one on one explanations to my supervisors were leaning much more into details and more generally what problem I wanted to tackle with a particular method.

This internship allowed me to develop my skills in working autonomously on a research project for a long time, notably on choosing which trails to explore and which to not. It made me learn me how to work with specific constraints (only local LLMs for instance), and to anticipate the monthly intern seminar presentation.

6.2 Culturally wise

Another aspect was the cultural differences with Europe, as I was led to interact with many Vietnamese people. While it did not change my vision of the world, it was nonetheless very interesting to speak with people having grown in a very different environment and therefore had a different relation to many things, as family or privacy. Discovering the vietnamese way of working and living was also very interesting since I do get the impression

that the heat is dictating the rhythm of their lives (a hearty breakfast at 6AM, a nap after lunch and bedtime at 9PM).

It is particularly impressive to see the lack of night life since almost every shops, restaurants or bars are closing at 11PM, and the streets are deserted at this hour. And amidst this, some little shops open all the nights are even more standing out since some people are keeping their shops open for one or maybe two purchase for the night. In the same way, it is remarkable to see that there is almost no unemployment, basically everybody has a job even if the utility of said jobs is questionnable. For instance in museum there is often one person to sell a ticket, and another to verify your ticket to let you enter the museaum 5 meters after that, with another a guard is present to watch the queue, which is the total opposite of what is happening in France where we are trying to automatize everything and reduce the number of employees.

7 CSR report of ACROSS

7.1 Overview

ACROSS Laboratory is attached to several organizations, the most important one being the french IRD, tackling problems in link with development in several countries. The core mission in ACROSS is the transmission of knowledge using modelling systems and Virtual Reality. The number of people in the laboratory is around 3 permanent french IRD employees, 5 vietnamese employees and a varying (2-4) PhD student, alongside invited researchers from diverse universities. ACROSS also employs numerous interns, there is almost always at least an intern, and at most a little under 10.

7.2 Governance and ethics

Transparency is an important value to the lab as researchers, so the code produced is made available on Github. There are some conflicts of interests since the laboratory is located inside an university, so the lab has to show to the rector the work that is being done, and sometimes there are visiting Chinese delegations in the lab. In the same way, every year the school of Thuy Loi is hosting a large (12-15 people) group of netherland student, who are working on an ongoing project with Thuy Loi, totally unrelated to ACROSS, and each year they are told to work inside the lab because they are also foreigners. And that is a weird decision since it basically overcrowd the laboratory and beside from lunch we have no reason to talk to each other, moreover their group is large enough that it doesn't "need" to integrate itself. Overall this decision for the university led the laboratory to be crowded with no interest for research purposes.

7.3 Environmental responsibility

This one is hard to judge because the laboratory being located on the other side of the globe necessarily causes multiple plane travels, even more so be-

cause of the high number of interns (10 interns a year = 20 Vietnam-France travels), and the delays for Vietnamese Visas often cause what is called a "Visa Run", ie going into another country to refresh a temporary visit Visa. This refresh is available only by plane, a lot of interns end up doing at least another fly to Bangkok. For the visiting researchers, the carbon cost is also high since they often work in France, so each visit is again 12H of plane travel. The environmental cost of the laboratory to exist is a little bit tricky to analyse since it is located in Vietnam, and therefore cannot be held to the same standards as a lab or enterprise in France. The AC is basically always on during day, and is necessary for health reasons. The electrical consumption of the lab is on the low side (as far as computer science labs are concerned) since the work here is mostly development and not production, the major consumption of this last 6 month is my work with local LLMs.

On the other side, the laboratory is actively participating and creating campaign of sensitization toward ecological issues for the Vietnamese children, and has developed several games to show the ecological impact of certain decisions.

7.4 Social and Human resources

The lab has been hiring both Vietnamese and French people, leading to a good social mixing. While the french people are mostly doing computer science while vietnamese people are mostly doing social science, there is some mixing and some french work in social science while some vietnamese work in computer science. As usual, the computer scientists are mostly mens, and the social science part is mostly composed of women. It is also of note that the laboratory is hiring artists for some projects, and is therefore open to other professions. We work in an open space with no problems of inclusivity or accessibility whatsoever.

7.5 Societal impact

A major project of the lab, SIMPLE, aims at providing serious VR games in order to help with children's education. In this light, the laboratory is undeniably contributing positively to the well-being of Vietnam's classrooms. Other serious games of flooding or ecology are also produced, meaning that a population more adult is also impacted. In the same way, the social science studies aims at improving the living conditions in Vietnam, and therefore are beneficial for the local population. ACROSS's tools have also helped the Vietnamese government to model the propagation of COVID-19 during the pandemic, helping the government to take decisions regarding which sectors or streets to lockdown.

ACROSS is also regularly publishing papers both in social science and in computer science and as said earlier, all of the produced code is available on github. The laboratory is therefore actively helping the scientific community, even more so that it is hosting several PhD students at all times.

7.6 Partial conclusion

Looking at all of these facts, we can say that ACROSS has a lot of long term goals with positive impact on both the environment and the local population, however the carbon footprint of the laboratory is quite high because of its policies of recruitment and its location. Being in Vietnam is mandatory for working toward sustainable development in Vietnam, and the recruitment policies are sort of forced by the fact that the laboratory is attached to the IRD, therefore we can say that all of these cost are a price paid to have a positive impact later.

8 Conclusion

My internship at ACROSS has been a very interesting one, both on a cultural and on a technical level. As an intern, I was given an almost total freedom on what methods I wanted to use, which has been very instructive on a lot of aspects. For one, it made me plan tasks and experiments ahead, forcing me to plan my research. Then, I also had to balance my time on thinking and my time of implementing on my own, which has been arduous since I much rather plan things than implement them, however this internship had forced me to balance my time and it also made me learn to not test an idea if that would mean too much time. The freedom of choosing which technologies to use was also very enjoyable since I could try new things, such as designing ways to graph a documentation, augment it and how to select pertinent things inside. I also have been in contact with many PhD student and interns with other background, which forced me to learn to vulgarize and adapt my speech to my interlocutor. It was also interesting on scientific culture basis since we were able to talk to each other about our work and I have been made aware of research very different from my field of expertise, which is always a good thing in my mind.

FAIRE AUSSI L'ALGO DE TRANSMISSION DE SIGNAL HOMOGENEISER LES NOTATIONS DES RELATIONS PRECISER LA GENERATION DE GRAPH

[1]

References

[1] Large Language Model Meets Constrained Propagation
Alexandre Bonlarron, Elisabetta De Maria, Florian Régin, Jean-Charles Régin

[2] Graph Retrieval-Augmented Generation: A Survey
BOCI PENG, School of Intelligence Science and Technology, Peking University, China YUN ZHU, College of Computer Science and Technology, Zhejiang University, China YONGCHAO LIU, Ant Group, China XIAOHE BO, Gaoling School of Artificial Intelligence, Renmin University of China, China HAIZHOU SHI, Rutgers University, US CHUNTAO HONG, Ant Group, China YAN ZHANG†, School of Intelligence Science and Technology, Peking University, China SILIANG TANG, College of Computer Science and Technology, Zhejiang University, China

[3] NodeRAG: Structuring Graph-based RAG with Heterogeneous Nodes
Tianyang Xu, Haojie Zheng, Chengze Li, Haoxiang Chen Yixin Liu, Ruoxi Chen, Lichao Sun

[4] Can Large Reasoning Models Self-Train?
Sheikh Shafayat, Fahim Tajwar; Ruslan Salakhutdinov, Jeff Schneider, Andrea Zanette