Translation of User Crochet Patterns to CrochetPARADE syntax using Large Language Models

Rachael Dias[1], **Kayvan Karim**[2]

[1] [2]Heriot-Watt University Dubai
rachaeldias123@gmail.com, K.Karim@hw.ac.uk

Abstract

Crochet, with its rich history and popularity, provides a creative and therapeutic outlet for millions across the globe, from many walks of life. However, crochet pattern creation and modification can be challenging for novice users, due to the spatial reasoning and structural understanding of stitches required. CrochetPARADE is a tool created to ease this process through pattern visualisation, but it uses a syntax that differs from standard notation and may not be intuitive to the average crocheter. This study explores the use of Large Language Models (LLMs) to translate user-generated crochet patterns into the CrochetPARADE syntax. The first structured, open-source collection of crochet patterns designed for machine learning applications was created, comprising user-generated patterns and their corresponding CrochetPARADE translations. Various approaches, including baseline, few-shot, and fine-tuning techniques, were evaluated with LLMs. The best results were achieved with fine-tuning DeepSeek-R1-Distill-Llama8b, reaching 74% accuracy, which has the potential to significantly improve the accessibility and ease of crochet pattern creation for users with varying levels of expertise.

Introduction

Crochet, a fibre art dating back to the 19th century, involves creating fabric swatches by interlocking yarn loops with a single hook (see Figure 1). Originally introduced in Ireland as a measure of famine relief (Britannica 2024), the craft has grown to over 28.8 million users in the USA alone, and spans a rather large demographic (AFCI 2016). As a result, crochet patterns (instructions to create crocheted items, as seen in Figure 3) are in abundance. Though these patterns are relatively easy to follow, translating this knowledge to create new patterns can be a difficult and daunting task for novice crocheters. This is due to the spatial reasoning required to visualise the shapes formed by crochet stitches, and an understanding of how the stitch structure can vary when created on top of another. There are attempts to simplify this process by introducing tools such as CrochetPARADE, a crochet pattern renderer, analyser and debugger. This software, however, uses a syntax that differs from a standard crochet pattern and may not be intuitive to an average crocheter. The aim of this project was to translate a user's crochet pattern to the syntax used by CrochetPARADE, to aid with standardisation and visualisation of the pattern.

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

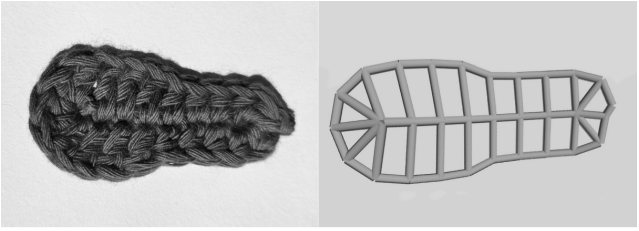
Figure 1: Process of creating a crochet stitch

Motivation

Beyond a hobby, crochet impacts personal wellness, social support, and microeconomics. It is also portable and low-cost (Burns and Van Der Meer 2020), making it accessible to individuals and broader communities. An international survey by Burns and Van Der Meer (2020) found that 89.5% of respondents felt calmer after crocheting, 82% experienced increased happiness, and 74.7% felt more productive. Many used crochet to manage mental health, grief, chronic illness, and pain. On a larger scale, the crochet community supports those in need. Organisations like Saaisha craft handmade breast prostheses and beanies for women and children undergoing cancer treatment, distributing 19,318 prostheses and 2,215 beanies as of September 2024 (Saaisha 2024). Crochet also enables financial independence, especially in areas with high unemployment. Iota, a sustainable furniture brand, trains women to crochet and offers jobs to help them gain autonomy (Peng 2024).

With such a significant impact worldwide, we seek to enable users to concentrate more on the creative process, rather than the repetitive manual work involved in crochet pattern creation and modification. This project, more specifically, aimed to:

- Create a dataset of user crochet patterns and manually translated CrochetPARADE patterns.
- Evaluate Large Language Models (LLMs) for translating crochet patterns into CrochetPARADE syntax using baseline testing and few-shot learning approaches.
- Fine-tune the selected LLMs on the created crochet pattern dataset.



Row 1: Ch 9, turn.	9ch , turn
Row 2: Sk the first stitch, (2 hdc in next st, 3 hdc, 3 dc, 5 dc in next st), turn.	sk , (hdc2inc , 3hdc , 3dc , dc5inc) , turn
Row 3: (1 dc in st attached to 2nd st in row 1, 2 dc, 3 hdc, 2 hdc in next st), ss in 2nd-to-last st in row 1.	dc@[0,1] , 2dc , 3hdc , hdc2inc) , ss@[0,-1]

Figure 2: Comparison of patterns: Natural Language Pattern with corresponding item (left), CrochetPARADE pattern with corresponding visualisation (right)

- Establish a strategy to evaluate model outputs for syntactic accuracy and pattern correctness.
- Conduct a final evaluation based on the framework to assess pattern translation.

Background

In this section, we formalise crochet patterns, introduce the CrochetPARADE software and its role in our project, and discuss developments with Machine Translation with a focus on the transformer architecture and Large Language Models (LLMs). We then criticise related works in the field and draw comparisons between them.

Crochet Patterns

Crochet patterns serve as essential guides, offering instructions to crocheters that are often divided into rows of interconnected stitches. However, crochet patterns for the same output item can be written in several ways, varying significantly in the skill level they are tailored to. More detailed patterns explicitly indicate stitch placement and provide full stitch names, while compact patterns rely on abbreviations and assume familiarity with techniques like turning, chains and stitch repetitions.

This variation presents specific challenges in pattern interpretation. Abbreviations and terminology can vary between pattern designers, with the same terms sometimes indicating different techniques. More complex instructions combine multiple operations in condensed formats, such as ”[sc, ch1] 5 times”. Despite these differences, the core structure of a pattern revolves around stitch types and their repetition counts. However, correctly following these instructions depends on recognising abbreviations and interpreting how condensed phrases expand into physical stitches.

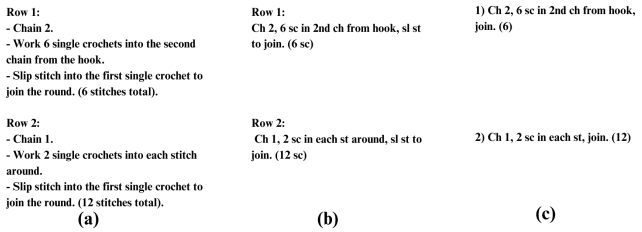


Figure 3: Comparison of equivalent crochet patterns written in natural language

CrochetPARADE and its Capabilities

The CrochetPARADE tool (Tassev 2023) enables users to create, visualise, and analyse crochet patterns in both 2D and 3D using a specialised grammar similar to a programming language. This ensures precision and avoids ambiguities common in plain English instructions. The code parses patterns for correctness, creates a virtual model, and renders it in 3D (see Figure 2). Users can interact with the model, adjusting rotation, zoom, yarn thickness, and color. The tool also supports exporting patterns and their output in formats such as plain text, SVG, and 3D files compatible with software like Blender (Tassev 2023). By incorporating pattern translation into the CrochetPARADE grammar, this highly useful tool could greatly increase its reach by eliminating the learning curve associated with using it. This task falls into the field of Machine Translation (MT), which focuses on using technology to automatically convert information from one format or language to another.

Large Language Models (LLMs)

MT breaks down language barriers on a global scale, enabling smoother communication and greater access to information (Benmansour and Hdouch 2023). Transformer-based models such as BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pre-trained Transformers), and more recently LLaMA (Large Language Model Meta AI), have introduced new levels of understanding, generation, and translation across a wide range of natural and formal languages (Zaki 2024).
BERT takes advantage of unsupervised pre-training with a bidirectional architecture, and so by analysing text in both directions, the model is able to tackle a large variety of Natural Language Processing (NLP) tasks (Devlin et al. 2019). Similarly, GPT demonstrates that improvements in NLP tasks can be achieved through pre-training of a language model on a varied collection of unlabeled text (Radford et al. 2018), followed by fine-tuning, the process of taking pre-trained models and training them further on specific data (Parthasarathy et al. 2024). The LLaMA models (Touvron et al. 2023) are especially suitable for research, with these state-of-the-art models trained using only publicly available datasets.

More recently, we see the introduction of reasoning models that make use of large-scale reinforcement learning

(RL). DeepSeek-R1 is a model trained using a multi-phase training approach that begins with a cold-start data stage before applying RL, significantly enhancing reasoning performance (DeepSeek-AI et al. 2025).

The potential for an LLM in translating crochet patterns arises from the combination of syntax and semantics within these instructions. These patterns involve specific syntax, such as stitch names along with their repetition indicators, as well as semantic elements, such as descriptions about stitch attachment or how to begin a row. A traditional compiler, which is a software generally used to translate code between languages, follows syntactic rules (Capon and Jinks 1988) and lacks the flexibility to interpret meaning beyond its predefined grammar. Additionally, some crochet patterns omit steps that experienced crocheters infer, posing challenges for more structured automation. Therefore, we make use of LLMs to capture both the syntax and pattern meaning and resolve the variations in crochet patterns that have the same output.

LLM Task Adaptation

Popular methods for adapting LLMs to domain-specific problems, including MT, include In-context Learning (ICL) and fine-tuning. ICL involves prompting a model with task demonstrations at inference time, without modifying the model's parameters. This method is popular due to its simplicity and improved generalisation to out-of-domain tasks. However, results from this approach have fallen short compared to state-of-the-art fine-tuned models (Brown et al. 2020). Fine-tuning, on the other hand, involves updating the weights of a pre-trained model with a target dataset. This method performs well on many benchmarks, but needs a large task dataset (Mosbach et al. 2021).

Parameter Optimisation Methods There are multiple ways to optimise fine-tuning that have been introduced. Among these is adapter-based fine-tuning, a family of efficiency techniques that involve freezing a pre-trained language model and introducing a small number of trainable parameters within its layers. This approach reduces training time with minimal effect on performance (Razuvayevskaya et al. 2024). Alternatively, Hu et al. (2022) proposed Low-Rank Adaptation (LoRA), a technique that freezes pre-trained model weights and adds trainable rank decomposition matrices into each layer. These approximate large weight matrices in a model by breaking them down into smaller matrices, reducing the number of parameters needed to adapt the model to new tasks. As seen in Figure 4, we only train *A* and *B*, the low-rank matrices, while keeping *W*, the pre-trained weights, intact.

MT Evaluation Metrics

There exist many metrics to evaluate the quality of machine-translated text. One commonly used (Han and Wong 2016) is BLEU (Bilingual Evaluation Understudy) (Papineni et al. 2002), measuring the similarity between a candidate translation and reference translation(s) based on matching n-grams (sequences of length "n").

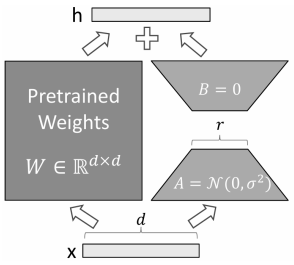


Figure 4: LoRA's reparameterization, only training A and B (Hu et al. 2022)

An alternative metric to BLEU is chrF (character n-gram F-score) (Popović 2015), a metric useful when word-level evaluation is insufficient. It focuses on character n-grams instead of word n-grams, providing a more precise measure of translation accuracy. chrF is computed by considering the character n-grams of both the reference and candidate translations. The formula for $CHRF_\beta$ is:

$$CHRF_\beta = (1 + \beta^2) \cdot \left(\frac{CHRP \cdot CHRR}{\beta^2 \cdot CHRP + CHRR} \right)$$

where $CHRP$ is the arithmetic average of character n-gram precision over all n-grams, $CHRR$ is the arithmetic average of character n-gram recall over all n-grams and β determines the importance of recall relative to precision.

Evtikhiev et al. (2022) found that chrF is the closest match to human assessment for code generation against multiple tested metrics, so it is useful for translation into CrochetPARADE syntax.

Related Work

Since there are no observed applications of MT specifically to crochet, we criticise works that fine-tune LLMs for translation tasks, focusing on their approach and methodology.

Alves et al. (2023) attempts to adapt LLMs to MT by using LoRA, as well as traditional full fine-tuning on the Llama 7B and 13B models. They found that adapter-based fine-tuning with LoRA performs as well as traditional fine-tuning, while reducing the number of training parameters by 50 times. Despite its increased efficiency, this research focuses on English-centric language papers which are in high quantity, and does not consider the generalisation to low-resource languages or non-English language pairs. This highlights a bigger challenge in MT research: adaptability across diverse domains.

To address this limitation, the Advanced Language Model-based trAnslator (ALMA) fine-tuning process was

introduced by Xu et al. (2023), and consists of two stages. The first involves improving the model's proficiency in non-English languages by fine-tuning on monolingual data. The second stage involves fine-tuning using a small set of high-quality parallel data. This technique, applied to Llama-2, found that LLMs do not require very extensive parallel data for MT. However, while ALMA does mark progress for LLMs translating in diverse domains, its performance remains inferior to models like GPT-3.5-T and GPT-4.

There are also fine-tuning methods such as Multilingual Fine-tuning with Translation Instructions (mFTI), a method to train LLMs to follow translation instructions directly as introduced by Li et al. (2023). This is done by organising translation tasks as different instances, where each instance is associated with a specific language pair and its corresponding instructions. mFTI performed better compared to ICL, where instructions are provided within the input prompt. This method, however, suffers from over/under translation, as well as hallucination, and emphasises the difficulty in both following instructions and maintaining context. Another instruction-based fine-tuning method is the framework ParroT, proposed by Jiao et al. (2023) that uses three types of instructions to enhance translation: translation instruction, contrastive instruction (by using different translations for a source sentence), and error-guided instruction. These translation instructions improve the translation performance of vanilla LLMs, and with LoRA, also prevents over-fitting to the data.

Even with these advancements, these translation solutions still struggle with high training costs for domain adaptation and the struggle to translate rare words in specific domains. To deal with this, Zheng et al. (2024) proposed LlamaIT, a prompt-oriented fine-tuning method. By fine-tuning Llama2-7B with a mix-domain dataset that is task-specific using LoRA, and incorporating vocabulary containing rare words into prompts, translation capabilities are greatly improved with less over-generation. However, as with other solutions, this research does not address its capabilities for low-resource languages.

Finally, Zhu et al. (2024) uses Supervised Fine-Tuning (SFT), with varying levels of labeled, parallel data, comparing them to ICL baselines and instruction-tuned LLMs. It was found that LLMs can be effectively fine-tuned for multilingual translation with as few as 32 parallel sentences. However, similar to Alves et al. (2023), this research is primarily focused on English-centric translations, highlighting the consistent gap in addressing truly multilingual or low-resource use cases.

Methodology

This section outlines the methodology for data collection, pattern translation, model selection, and fine-tuning. We also detail the dataset construction process to ensure a representative and scalable foundation, and explain the selection and evaluation of large language models (LLMs) through baseline inference, few-shot learning, and fine-tuning.

Data Collection

The dataset construction process began with learning the CrochetPARADE manual, which covers information about the platform as well as its specialised grammar. Once familiar with the manual, we next analysed the features present within this language. This involved identifying core components such as stitch names, how repetitions are represented, increases, decreases, and more. We then created subsets of features of the language to add to the dataset incrementally, ensuring a structured approach to dataset expansion. These subsets are represented in Figure 5, illustrating the progressive inclusion of more complex features. From this, set A and set B were selected for inclusion in the current dataset, prioritising those that appear most frequently in user patterns and those that serve as a good base for future expansion.

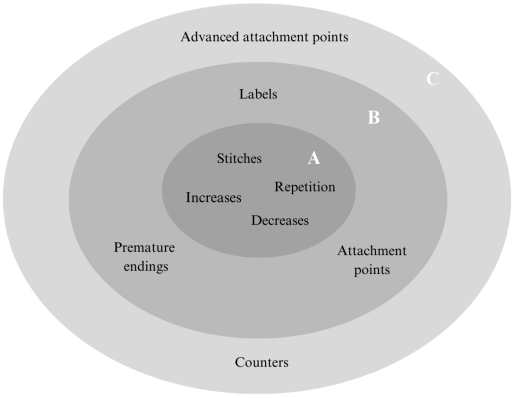


Figure 5: Structured subsets of CrochetPARADE features for incremental dataset expansion

We then required a collection of crochet patterns written in natural language. This collection process was conducted in two ways: writing original patterns and extracting sections from publicly available patterns online. The written patterns were designed to cover a range of features and structures seen in the CrochetPARADE language, ensuring diversity in the dataset. Meanwhile, the gathered patterns were collected to maximise variety in the dataset while resembling patterns that exist online. However, as crochet patterns are the intellectual property of their designers, only patterns in the public domain were used for this project. This means they are no longer under the protection of copyright, and users are free to copy, share, and sell these patterns. To ensure compliance, all external patterns were sourced exclusively from the following websites:

- https://freevintagecrochet.com
- https://antiquepatternlibrary.org

Data Translation and Compilation

For each of the collected patterns, we have used the CrochetPARADE manual as our translation guide. We went row-by-row through the current pattern, assessing the stitches and features that are used, and then manually

translated them. After each pattern is translated, we ran the pattern in the CrochetPARADE software to verify its syntactic correctness, and then compared the 3D-rendered output to images in the user pattern, if provided. The manual thoroughly covers different aspects of the grammar, and so the combination of this manual and software helped build an effective translation process. All patterns were compiled into a CSV file that contains the original pattern, the translated pattern, and other data, such as the shape it is meant to represent. We also assigned classes to each of the translated patterns, representing features from subsets A and B in Figure 5, and have described these classes in Table 1. A combination of letters (e.g., PL) indicates the presence of features from multiple classes within a single pattern. Additionally, we assume that all patterns inherently include elements from Subset A (Class B).

The final dataset contained 109 patterns. The dataset CSV file was split into eight train and test files, taking approximately 1/8 of each class for testing, ensuring no overlaps between them, using a Python script. A validation set was not created due to the limited size of the dataset.

Initial Testing and LLM Selection

We selected suitable open-source models to test from various sources, including models used in translation problems, as described in our Related Work. Since we are using Unsloth for fine-tuning, we also considered popular models on the platform based on likes and downloads. In addition, DeepSeek has contributed to the research community by open-sourcing six dense models distilled from DeepSeek-R1 based on Qwen and Llama (DeepSeek-AI et al. 2025). From these, we selected DeepSeek-R1-Distill-Llama-8B for our experiments. With this, our chosen models were as follows:

- Llama3.2-3b
- Llama3.1-8b
- Deepseek-R1-Distill-Llama-8b
- Qwen2-7b
- Mistral-7b

LLM Pattern Generation Approaches

We evaluated three approaches for generating crochet patterns using LLMs: baseline inference, few-shot learning, and fine-tuning. To do this, we make use of Unsloth, a library optimised for efficient model training and fine-tuning which supports full fine-tuning, pretraining, and low-bit training (4-bit, 8-bit, and 16-bit). Before applying any LLMs, the eight training CSV files were first converted into a question-answer JSON format.

Baseline Testing Baseline inference was carried out for all models by loading the pre-trained model and tokeniser, processing the dataset of questions and expected answers, and generating responses using the model. The only difference across models was the use of different chat templates, with each model following the same general inference procedure.

Class	Name	Description
B	Base Pattern	Features from Subset A. Basic elements such as stitch names, repetition of a stitch, increasing and decreasing within a single stitch.
L	Labels	Marking a stitch, or group of stitches, to work into at a later point (eg. 2sc**.A**, where .A is defining the label)
A	Attachment Points	Using the @ symbol to work into a specified stitch or into a defined label (eg. sc**@A**, working a single crochet into the defined label)
P	Premature Endings	Condensing a long string of repeating stitches into a shorter format (eg. [2sc,>,dc]*3 is equivalent to 2sc,dc,2sc,dc,2sc)

Table 1: Class Descriptions

Few-shot Testing To conduct few-shot testing, we first modified the input prompts in the dataset to provide the models with context. Specifically, we chose three examples that encapsulate all features of our dataset well and appended them to all input patterns using a Python script. Following this, we used the same steps as in the baseline testing.

Model Fine-tuning To fine-tune each model, the model and tokeniser are first loaded, and LoRA is applied to enable parameter-efficient fine-tuning. The training data is preprocessed using a chat template, and then training is configured with appropriate hyperparameters like batch size and learning rate. After training, inference is run on test data, and results are saved. This process is repeated for each split to iteratively fine-tune the models.

Evaluation Strategy and Results

The evaluation of translated user-generated crochet patterns into the CrochetPARADE grammar focuses on both syntactic and semantic criteria to ensure that the translated patterns accurately reflect the intended structure and design of the user's original pattern. When calculating accuracy, we found that automating a direct comparison to identify exact pattern matches was not feasible. This is because minor variations can exist without affecting the final pattern output. For example, assigning a label as "A", "B", or any other letter has no impact on the pattern, and adding unnecessary parentheses around lines does not alter the outcome either. So, we manually compared over 1500 patterns to observe these variations and compare the 3D-rendered outputs of the manually translated patterns to those generated by our model to measure accuracy. This comparison involved a stitch count and

an examination of stitch types, which can be accessed in the 3D view in CrochetPARADE by hovering over each stitch. We define syntactic correctness as the number of patterns that execute successfully in the CrochetPARADE platform, without generating syntax errors. In addition to the classes described in Table 1, we also test Class LA, which are patterns that contain both features from Class L and Class A.

Baseline Testing

When performing baseline testing, since none of the tested models were trained on CrochetPARADE syntax and had no knowledge of the platform, their outputs were syntactically incorrect and therefore did not match any of the original translated patterns. Instead, the models often attempted to define CrochetPARADE itself, sometimes guessing at possible abbreviations rather than generating usable instructions. For example, outputs from Llama3.2-3b incorrectly define repetitions using 'REP' instead of the language's bracket notation. Meanwhile, we also see models like Deepseek-R1 invent expansions for the platform's name, such as "Pattern Authoring Rules And Diagram Encoding". Errors like these reveal that without explicit training on CrochetPARADE, these LLMs default to approximation (or fabrication) of the language and its rules.

Few Shot Testing

Tables 2 and 3 reveal significant improvements in accuracy and syntactic correctness, respectively, compared to our baseline testing, when using ICL. For accuracy, Qwen2-7b leads (24.3%) with a strong performance in Class P (50%), while Llama3.1-8b also performs its best in Class P (37.5%). For syntactic correctness, Qwen2-7b again performs best (40.8%) with particular strength in Class A (62.5%), and Deepseek-R1 emerges as the only model achieving any correctness in Class L (25%). In addition, all five models score 0% accuracy in Class L and LA, showing class-specific struggles. The overall accuracy is depicted in Figures 6.

Similar to our baseline testing, we see models defining stitches on their own. For example, we see Mistral-7b defining the slipstitch as 'slst' instead of the correct 'ss' on multiple occasions. We also see interesting output from our reasoning model, Deepseek-R1, within its <think></think> tokens. In more than one example, we see the output reasoning through each example pattern provided and by the end, outputs "Wait, I'm getting confused. Let me check the examples again." and repeats the same process. It also produced unexpected terms for some patterns, for example, outputting "double crocodiles" instead of "double crochet". These results together demonstrate that while few-shot learning enables some syntactic validity, models still lack consistent, accurate pattern translation.

[1]All mentions of DeepSeek-R1 from this point are referring to Deepseek-R1-Distill-Llama-8b.

Model	A	P	L	LA	B	O
Llama3.2-3b	4.2	0	0	0	13.7	8.7
Llama3.1-8b	12.5	37.5	0	0	33.3	22.3
Mistral-7b	8.3	8.3	0	0	29.4	18.4
Qwen2-7b	33.3	50	0	0	21.6	24.3
Deepseek-R1[1]	16.7	0	0	0	23.5	15.5

Table 2: Fewshot Comparison of LLMs by Accuracy for Each Class and Overall Accuracy (O) in %

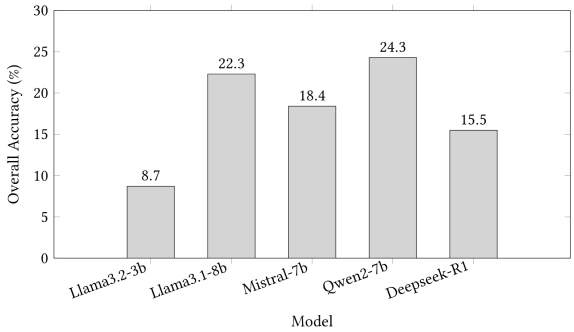


Figure 6: Bar Chart for Fewshot Comparison of LLMs by Overall Accuracy

Fine-tuning Results

Tables 4 and 5 show the results of the fine-tuning with accuracy and correctness, respectively, with 8-fold cross-validation. DeepSeek-R1 achieved the highest overall accuracy at 74%, followed by Qwen2-7b (70.2%), Mistral-7b (69.2%), and Llama3.1-8b (69.2%), with Llama3.2-3b performing the lowest at 46%. The overall accuracy is depicted in Figure 7. Interestingly, while Qwen2-7b had the highest syntactic correctness (87.5%), its overall accuracy (70.2%) suggests it may generate syntactically correct but incorrect outputs. Conversely, DeepSeek-R1, despite having the highest accuracy, did not achieve the highest correctness, though it was not far behind. This could indicate that patterns it translated incorrectly were also less likely to be structured correctly as valid CrochetPARADE patterns. Nevertheless, with its strong overall performance, DeepSeek-R1 remains the most effective model in this evaluation.

For patterns that are syntactically correct but do not accurately reflect the original pattern, we examine what kind of variations occur. For instance, Figure 8 visualises a pattern from the dataset where the output takes the shape of a cone. The output pattern generated by Llama3.2-3b is shown in Figure 9, where, despite maintaining the general struc-

Model	A	P	L	LA	B	O
Llama3.2-3b	12.5	25	0	12.5	19.6	17.5
Llama3.1-8b	45.8	37.5	0	37.5	37.3	35
Mistral-7b	37.5	8.3	0	25	41.2	34
Qwen2-7b	62.5	50	0	50	29.4	40.8
Deepseek-R1	25	25	25	37.5	25.5	24

Table 3: Fewshot Comparison of LLMs by Correctness for Each Class and Overall Correctness (O) in %

Model	A	P	L	LA	B	O
Llama3.2-3b	25	0	10	14.3	83.7	46
Llama3.1-8b	70.8	37.5	40	14.3	91.8	69.2
Mistral-7b	66.7	50	60	14.3	89.8	69.2
Qwen2-7b	66.7	62.5	0	28.6	91.9	70.2
DeepSeek-R1	75	62.5	57.1	42.9	87.8	74

Table 4: Fine-tuning Comparison of LLMs by Accuracy for Each Class and Overall Accuracy (O) in %

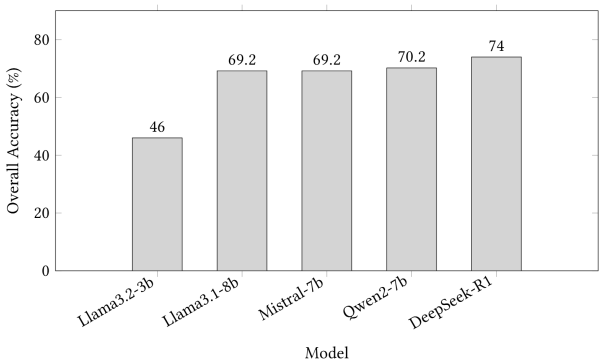


Figure 7: Bar Chart for Fine-tuning Comparison of LLMs by Overall Accuracy

Model	A	P	L	LA	B	O
Llama3.2-3b	70.8	37.5	40	28.6	87.8	68
Llama3.1-8b	83.3	62.5	80	28.6	91.8	79.8
Mistral-7b	87.5	87.5	80	28.6	95.9	86.5
Qwen2-7b	95.8	75	50	100	91.9	87.5
DeepSeek-R1	87.5	75	71.4	42.9	91.9	82.5

Table 5: Fine-tuning Comparison of LLMs by Syntactic Correctness for Each Class and Overall Correctness (O) in %

ture (creating a ring and working into it), a single crochet is used instead of a triple crochet. Since the single crochet is a shorter stitch, the output appears more circular than conical. This shows how small syntactic changes can significantly impact the final shape of a crochet pattern.

chrF Evaluation

On our most successful pattern generation method, fine-tuning, we continue evaluating the model output. In order to compare the matching character n-grams between the original and translated pattern, we applied the chrF metric, where a score of 0 means no overlap and 1 means a perfect match at the character level between the generated and reference translation. The results, shown in Figure 10, indicate that DeepSeek-R1 achieved the highest average chrF score at 0.851, suggesting it produces the most character-level accurate translations overall. Qwen2-7B and Mistral-7B followed closely with scores of 0.839 and 0.837, respectively, demonstrating their strong ability to generate outputs that align well with the reference patterns. Llama-8B also performed well with a score of 0.819, while Llama-3B had the lowest chrF score at 0.737, indicating a comparatively lower character-level similarity.

Interestingly, while DeepSeek-R1 did not have the highest correctness score, its high chrF score suggests that its outputs are still very close to the reference patterns at the character level. This implies that the patterns it generates, even when incorrect, may not require significant modifications to become valid, reinforcing its strong overall performance.

Error Analysis

For patterns that fail to execute due to syntax errors, an error analysis was conducted. This analysis categorised the types of syntactic errors that occur as a pop-up on the CrochetPARADE platform, such as missing labels or undefined stitches. Identifying and categorising these errors allows us to draw conclusions about areas where the translation model may be prone to misinterpretation.

The most common error across models was ”Label not Found”, which occurs when working with an attachment point that references a label that was never defined. Another frequent error, primarily seen with Llama3.2-3b, was ”Stitch not Defined”. This happens when the model generates stitch names that are not recognised by the CrochetPARADE platform, such as using ”slst” instead of ”ss” for a slip stitch or ”tc” instead of ”tr” for a triple crochet. Additionally, the ”ID not Found” error appeared across all fine-tuned models. This happens when the number of stitches in a row exceeds

ring.R
6tr@R
(2tr2inc)*6

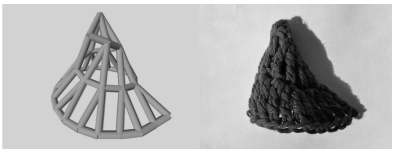


Figure 8: Cone Pattern with correct output, in CrochetPARADE and crocheted

ring.R
sc6inc@R
sc2inc*6

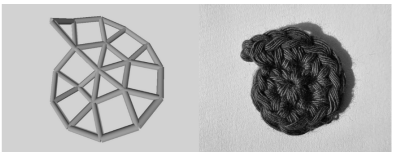


Figure 9: Cone Pattern with Llama3b output, in CrochetPARADE and crocheted

the available foundation stitches from the previous row, leaving some stitches without a valid placement. Essentially, the model is trying to work into an ID that doesn't exist. These errors highlight recurring issues in pattern structure and terminology, providing clear targets for refining the dataset and training process.

Conclusion

This project aimed to improve the creation and modification of crochet patterns by evaluating an LLM's ability to translate user-written patterns into the CrochetPARADE syntax. The key objectives were to (1) create a dataset of user-generated patterns and their translations, (2) evaluate LLMs for translation with and without context, (3) fine-tune selected LLMs, (4) develop an evaluation framework, and (5) assess model performance.

Significant progress was made. The dataset represents the first structured, open-source collection of crochet patterns for machine learning, with over 100 patterns spanning various techniques. The best-performing model, DeepSeek-R1-Distill-Llama8b, achieved 74% syntactic accuracy. Fine-tuning on a domain-specific dataset led to a 58.5% improvement over few-shot learning. The evaluation framework combines automated checks with expert review and offers a methodical way to assess pattern translations. These contributions help bridge computational methods and traditional craft knowledge.

Limitations and Future Work

Some limitations remain. Accuracy varied with pattern complexity, and models struggled with classes like L ("Labels"), often referencing labels before defining them. The dataset, while diverse, does not cover the full range of crochet features. Manual validation, though necessary, limited scalability.

Future work should expand the dataset, especially with patterns in Subset C in Figure 5. Syntactically correct but semantically inaccurate patterns could be reverse-engineered to add more examples. Further model improvement through reinforcement learning and real-time user feedback could boost translation quality. A user-facing system for validating outputs would also support continuous refinement.

This project lays a foundation for automated crochet pattern translation and highlights both the potential and challenges of this task. Addressing current gaps will help make patterns more accessible and standardised.

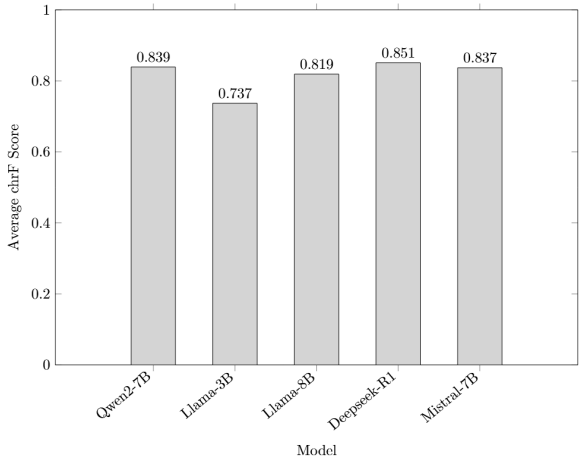


Figure 10: chrF results across models

The created dataset can be found at: https://github.com/rachaelteresa/StitchSwitch

References

AFCI. 2016. Study of the US Craft Market. Accessed: October 15, 2024.

Alves, D. M.; Guerreiro, N. M.; Alves, J.; Pombal, J.; Rei, R.; de Souza, J. G. C.; Colombo, P.; and Martins, A. 2023. Steering Large Language Models for Machine Translation with Finetuning and In-Context Learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 11127–11148. Singapore: Association for Computational Linguistics.

Benmansour, M.; and Hdouch, Y. 2023. The Role of The Latest Technologies in the Translation Industry. *Emirati Journal of Education and Literatures*, 1(2): 31–36.

Britannica, E. 2024. Crochet. Accessed: October 15, 2024.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; and Kaplan. 2020. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 33: 1877–1901.

Burns, P.; and Van Der Meer, R. 2020. Happy Hookers: Findings from an International Study Exploring the Effects of Crochet on Wellbeing. *Perspectives in Public Health*.

Capon, P.; and Jinks, P. 1988. *Introduction — What is a compiler?* Palgrave, London.

DeepSeek-AI; Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; Zhang,

X.; and Yu, X. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Association for Computational Linguistics.

Evtikhiev, M.; Bogomolov, E.; Sokolov, Y.; and Bryksin, T. 2022. Out of the BLEU: how should we assess quality of the Code Generation models? *arXiv preprint*. Submitted on 5 Aug 2022 (v1), last revised 10 May 2023 (v2).

Han, A. L.-F.; and Wong, D. F. 2016. Machine Translation Evaluation: A Survey. *arXiv preprint arXiv:1605.04515*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Jiao, W.; Huang, J.-t.; Wang, W.; He, Z.; Liang, T.; Wang, X.; Shi, S.; and Tu, Z. 2023. ParroT: Translating during Chat using Large Language Models tuned with Human Translation and Feedback. *arXiv preprint arXiv:2304.02426*.

Li, J.; Zhou, H.; Huang, S.; Cheng, S.; and Chen, J. 2023. Eliciting the Translation Ability of Large Language Models via Multilingual Finetuning with Translation Instructions. *arXiv preprint arXiv:2305.15083*.

Mosbach, M.; Pimentel, T.; Ravfogel, S.; Klakow, D.; and Elazar, Y. 2021. Few-shot Fine-tuning vs. In-context Learning: A Fair Comparison and Evaluation. Mmosbach@lsv.uni-saarland.de.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311–318. Philadelphia: Association for Computational Linguistics.

Parthasarathy, V. B.; Zafar, A.; Khan, A.; and Shahid, A. 2024. The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities. Technical report, CeADAR: Ireland's Centre for AI, University College Dublin, Belfield, Dublin, Ireland. @ CeADAR Connect Group.

Peng, J. 2024. *The Slow Fashion Renaissance: An In-Depth Exploration of Crochet and Knitting as Sustainable Technologies for Contemporary Fashion*. Master's thesis, Politecnico di Milano.

Popović, M. 2015. CHRF: character n-gram F-score for automatic MT evaluation. Maja.popovic@hu-berlin.de.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving Language Understanding by Generative Pre-Training. *OpenAI Blog*.

Razuvayevskaya, O.; Wu, B.; Leite, J. A.; Heppell, F.; Srba, I.; Scarton, C.; et al. 2024. Comparison between parameter-efficient techniques and full fine-tuning: A case study on multilingual news article classification. *PLoS ONE*, 19(5): e0301738.

Saaisha. 2024. Saaisha India Foundation. Accessed: October 15, 2024.

Tassev, S. 2023. CrochetPARADE: Crochet PAttern Renderer, Analyzer, and DEbugger. Accessed: October 29, 2024.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; Rodriguez, A.; Joulin, A.; Grave, E.; and Lample, G. 2023. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*.

Xu, H.; Kim, Y. J.; Sharaf, A.; and Awadalla, H. H. 2023. A Paradigm Shift in Machine Translation: Boosting Translation Performance of Large Language Models. *arXiv preprint arXiv:2309.11674*.

Zaki, M. Z. 2024. Revolutionising Translation Technology: A Comparative Study of Variant Transformer Models - BERT, GPT and T5. *Computer Science & Engineering: An International Journal (CSEIJ)*, 14(3).

Zheng, J.; Hong, H.; Wu, S.; Wen, J.; and Wu, S. 2024. Fine-tuning Large Language Models for Domain-specific Machine Translation. *arXiv preprint arXiv:2402.15004*.

Zhu, D.; Xie, X.; Xia, Y.; Qin, T.; and Liu, T.-Y. 2024. Fine-Tuning Large Language Models to Translate: Will a Touch of Noisy Data in Misaligned Languages Suffice? *arXiv preprint arXiv:2404.14122*.