Fine-tuning Large Language Models for Automatic Font Skeleton Generation: Exploration and Analysis

Yuxuan Liu[1], Yasuhisa Fujii[2], Xinru Zhu[1], and Kayoko Nohara[1]

[1] School of Environment and Society, Institute of Science Tokyo, Tokyo, Japan
{liu.y.cd, zhu.x.ac, nohara.k.aa}@m.titech.ac.jp
[2] Google DeepMind, Tokyo, Japan
yasuhisaf@google.com

Abstract. Despite the pivotal role that font skeletons could play in typeface research and font design, the availability of font skeleton data is sparse and limited. This research explores the possibility of using Large Language Models (LLMs) to generate font skeleton data based on font outline data. Our method represents font skeletons and font outlines as sequences of text tokens derived from SVG commands, and formulates the font skeleton task as a language modeling task predicting the token sequence for a font skeleton given the token sequence of a font outline. As a first attempt, we fine-tuned GPT-3.5 on a dataset of 8,213 Japanese font outlines and corresponding skeletons. Both quantitative and qualitative evaluations show the effectiveness of the approach in terms of rasterized pixel distance, Chamfer distance, and visual analysis. Our proposed method achieved average results of 14.678 for rasterized pixel distance and 1.713 for Chamfer distance, both better than the baseline method (PolyVectorization). In visual analysis, we found better generation results for complex shapes which logograms such as Chinese characters tend to have, than for the simple shapes of syllabograms such as Japanese kana, phonograms such as Latin alphabets, and symbols. Although our fine-tuned model has limitations in generating the skeletons of other font styles, this research establishes a foundation for the automatic generation of font skeletons using LLMs, setting the stage for future work on automatic skeleton generation and the wider application of font skeletons in typography.

Keywords: Typeface design · Font skeleton · Automatic generation · Large Language Model · Fine-tuning.

1 Introduction

The emergence of digital typography over the past decades has seen a proliferation of font data formats, each striving to address the evolving needs of typeface designers [16, 17]. Among various options, outline font formats, which are implemented as OpenType or TrueType format, are the prevailing choice in contemporary computing environments [18].

The outline font formats are based on the scalable vector graphic which utilizes Bezier curves to represent scalable character outlines (see Fig. 1). Although they have overcome some of the limitations imposed by the preceding bitmap font format, such as the lack of scalability and the necessity for distinct fonts for each size, the process of type design remains time and effort consuming which involves the arrangement of strokes to shape letters and characters [28].

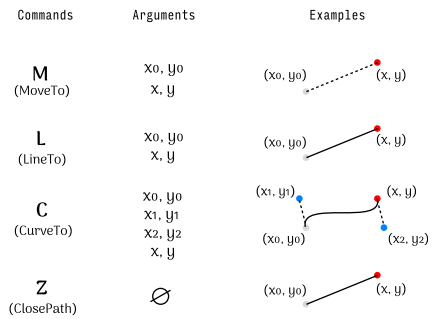

Fig. 1: An example of scalable vector graphic drawing commands

A stroke, which is composed of an outline and a skeleton (see Fig. 2), plays a pivotal role in type design, defining the visual attributes of characters [5]. While a font outline captures the details of a glyph through lines and curves, a font skeleton represents the fundamental structure of the glyph.

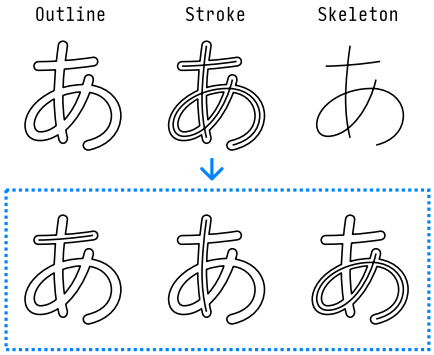

Fig. 2: Font outline, font stroke, font skeleton (the stroke encompasses both the outline, which defines the shape of the letters, and the skeleton, which is the underlying structure of the characters)

Designers usually draw font outlines directly when designing letters. However, previous research [10, 13, 14] and recent practice [19, 29] both suggest that font skeletons can be used as a basis for generating font outlines by applying specific rules and algorithms. This approach streamlines the type design process and offers designers the flexibility to modify and iterate on the outline generation based on the underlying font skeleton. Nonetheless, the availability of font skeleton data is sparse and limited.

To address the issue of insufficient font skeleton resources and lack of automatic generation methods of font skeleton data, this paper aims to explore the possibility of utilizing Large Language Models (LLMs) to automatically generate font skeleton data from existing font outline data.

As a first attempt, we will adopt fine-tuning, which is a technique that involves retraining pre-trained models on task-specific datasets [15, 27], on GPT-3.5 [8]. We will first fine-tune a GPT-3.5 model with a dataset consisting of 8,213 pairs of Japanese font outlines and skeletons. Then, we will generate new font skeletons with the fine-tuned models. Finally, we will evaluate the quality of the generated skeletons both quantitatively and qualitatively.

2 Related Work

2.1 Supervised Fine-Tuning

Supervised fine-tuning (SFT) has gained significant attention due to its effectiveness in adapting pre-trained language models for various downstream tasks. LLMs undergo SFT to enhance their performance in solving tasks and better align with human instructions.

Devlin et al. [11] introduced the BERT model, which demonstrated the power of fine-tuning language models for tasks such as text classification [12], named entity recognition [30], and question answering [23]. Numerous studies have further explored the application of SFT with variations of architectures such as InstructGPT [21], Imitation models [31], and Generative Pre-trained Transformer (GPT, GPT2) [25, 26].

In Brown et al. [8], GPT-3's capability to achieve high performance in diverse NLP tasks was highlighted with minimal exposure to training data. The improved task-agnostic and few-shot performance suggests its potential application of fine-tuning for more specific tasks. Related research has demonstrated the adaptability of GPT-3 through fine-tuning it for a diverse range of tasks, such as CodeX [9], mathmatical reasoning [32], and text summarization of low-resource languages [1].

The insights from these studies suggest that SFT could be applied for various language model problems. In this paper, we formulate the font skeleton generation as a language model problem and explore fine-tuning GPT-3.5 to build a task-specific model.

2.2 Font Skeleton Generation

Previous work of font skeleton generation can be divided into two categories: manual generation and automatic generation.

Manual Generation of Font Skeletons KanjiVG [3] is a project which exemplifies manual generation of font skeletons. It provides vector format font skeleton data as Scalable Vector Graphic (SVG) files with additional information, such as stroke order, about kanji characters (logograms) used in the Japanese language. Apel & Quint built the dataset aiming at compiling a graphetic dictionary for Japanese kanji characters [4, 24]. The authors created font skeleton manually based on existing Japanese typefaces in this project.

The manual procedures involved in creating font skeletons have a certain implications to automatic font skeleton generation.

Automatic Generation of Skeletons and Font Strokes As mentioned in Section 1, the methods for automatic font skeleton generation are sparse. Data-driven methods are extremely rare due to a limited amount of available data.

However, there are algorithm-based methods for extracting and vectorizing center lines of hand-drawn illustrations or writing trajectory which can potentially be applied to font skeleton generation, including PolyVectorization [7], which was proposed based on state-of-the-art mathematical algorithms for frame field processing.

In the context of type design, Berio et al. [6] proposed an algorithm-based method that employs shape analysis to automatically segment shapes into a set of overlapping and intersecting strokes. Although the algorithm is not directly targeting font skeleton generation, separated strokes can be served as a source for more optimized font skeleton generation.

Unlike data-driven methods, algorithm-based methods do not require an extensive dataset of font skeletons for training, but they do necessitate adjustments to parameters for specific shapes and additional rules across various cases and complex characters.

This paper proposes a data-driven method with a moderate number of data to address the obstruction of existing data-driven and algorithm-based methods.

3 Data

We use a dataset containing 8,213 pairs of font outlines and skeletons in this paper. The outlines were sourced from DNP Shuei MGothic Std L and the skeletons were manually extracted from the outlines. We show examples of the data in Fig. 3.

For the use in this study, we first preprocessed the outlines and skeletons to SVG format. The path information quoted in '`d="M ..."`' in SVG file were extracted as the main training context (see Fig. 4). The graphic scale and coordinates in the path were set at 50.0×50.0 with one decimal place precision. It

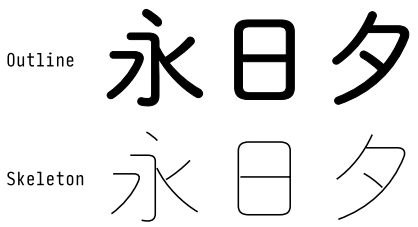

Fig. 3: DNP Shuei MGothic Std L (outline and skeleton)

was kept in this range for two reasons. Firstly, accumulating costs for generating long path information with the fine-tuned GPT-3.5 necessitated that we kept the path information short. Also, keeping the coordinates in same decimal place can avoid additional loss in language model learning.

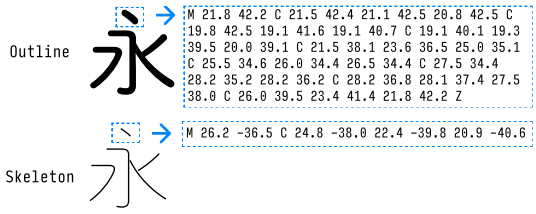

Fig. 4: Commands in SVG file (outline and skeleton)

Additionally, the characters in three different fonts (M Plus Rounded 1c, Noto Sans, and Noto Serif) of different font styles (rounded sans serif, san serif, and serif) were extracted and preprocessed for the purpose of additional qualitative evaluation. These fonts are demonstrated in Fig. 5.

The data is used in accordance with the license of each font.

4 Methods

4.1 Problem Formulation

In order to apply the vector images to the task of generating large language models, we propose a solution for transforming vector images into textual representations.

Considering that the path elements in vector images are determined by a number of commands (`M`, `L`, `C`, `Z`) and the coordinates of the points that commands are assigned to, we can treat the combination of these commands and numbers as a form of text contents. As we mentioned in Section 3, "`d`" attribute

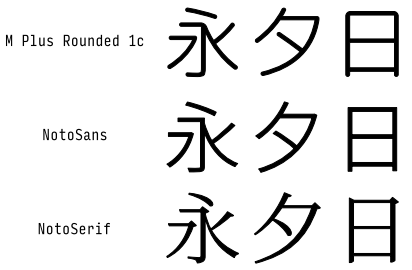

Fig. 5: M Plus Rounded 1c, Noto Sans, and Noto Serif (outline)

including commands and coordinates in a SVG format file defines a path to be drawn. We extract the "`d`" attribute and round the numbers of the coordinates to the first decimal place. We employ Python scripts to manipulate the font SVG files and render them into textual representations (see Fig. 4).

The textual representations of vector images of font outline and font skeleton were utilized as training/validation data for fine-tuning GPT-3.5 and testing data for the generation of skeleton.

4.2 Fine-Tuning GPT-3.5

Table 1: Fine-tuning parameters for GPT-3.5

Parameters	**Value**
Batch Size	10
Learning Rate Multiplier	2
Epochs	3

In this research, GPT-3.5, specifically version `gpt-3.5-turbo-0613`, was fine-tuned with training data of pre-processed font outlines and font skeletons using OpenAI's Python API [20]. All data was used to create prompts in JSONL format as per the API documentation. The font outlines were set as the `user`, and the font skeletons were set as `assistant`. We use the following format: `{"messages": [{"role": "user", "content": "M 5.7 7.9 C 5.7 3.8 8.1 1.7 11.7 1.7 ...(outline SVG commands)"}, {"role": "assistant", "content": "M 33.0 -33.5 C 26.9 -32.5 17.3 -31.6 11.1 -31.6 ...(skeleton SVG commands)"}]}` for fine-tuning and `{"messages": [{"role": "user", "content": "M 5.7 7.9 C 5.7 3.8 8.1 1.7 11.7 1.7 ...(outline SVG commands)"}]}` for inference. All hyperparameters set for the fine-tuning are detailed in Table 1. Training a single model typically took 4 hours 26 minutes 40 seconds, with an average financial cost of 248.44 USD.

5 Experiments

5.1 Font Skeleton Generation

We employed cross validation with 5 folds utilizing `KFold` in `scikit-learn` [22]. The validation data is further held out from the training portion in each fold with the ratio of 0.2.

The fine-tuned GPT-3.5 was applied to generate font skeletons from testing data. Since the limit of tokens for `gpt-3.5-turbo-0613` is 4096, complex characters whose path information exceeded the limit were failed to generate. For these cases, we set the path information in generation results to 'M 0.0 0.0 C 0.0 0.0' to represent a blank character. We identified 48 such cases in group 1, 29 cases in group 2, 41 cases in group 3, 47 cases in group 4, and 44 cases in group 5.

In addition, minor corrections were made to the generation results to fix invalid SVG structures. There are two types of such corrections: (1) missing elements in path such as commands and points and (2) redundant elements in path such as ',' and '"'. We identified 14 such cases in group 1, 43 cases in group 2, 41 cases in group 3, 57 cases in group 4, and 39 cases in group 5.

Furthermore, we used the fine-tuned model `gpt-3.5-turbo-0613-2` to generate skeletons of three additional fonts, namely M Plus Rounded 1c, Noto Sans, and Noto Serif, that we mentioned in Section 3. The testing data for group 2 were used for the experiment.

5.2 Quantitative Evaluation

Two evaluation methods are employed to measure the dissimilarity between two images: rasterized pixel distance, which quantifies the dissimilarity between raster images, and Chamfer distance, which assesses the dissimilarity between vector images [2].

Rasterized Pixel Distance Rasterized pixel distance (Eq. 1) is the distance between the generated and ground truth images in raster space. In order to calculate the distance, we rasterized the generated and ground truth skeleton data by (1) filling the paths in black with 1 stroke width in SVG and (2) using `cairosvg` to convert each skeleton data into a 50×50 pixel raster image.

Mean Absolute Error (MAE with L1 distance) of the luminance values is calculated as the Rasterized Pixel Distance [2].

$$\mathrm{RPD} = \mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{1}$$

where:

x_i : luminance value of the ith pixel of the ground truth
y_i : luminance value of the ith pixel of the generated image
n : number of pixels

Chamfer Distance Chamfer distance (Eq. 2) is a metric used to evaluate the distance between two sets of points.

$$\mathrm{CD}\left(S_1, S_2\right) = \frac{1}{|S_1|} \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \frac{1}{|S_2|} \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2 \tag{2}$$

where:

S_1 : a set of points in the ground truth
S_2 : a set of points in the generated image
x : a point in the ground truth
y : a point in the generated image

In our evaluation, Chamfer distance is computed for each image, treating all paths collectively as a group of points within one character. The number of point samples per command ('C' for curves, 'L' for lines) is set to 99, as per the equation employed for sampling points [2].

Results The results of quantitative evaluation are summarized in Table 2.

Table 2: Evaluation results by 5-fold cross-validation. RPD and CD stands for rasterized pixel distance and Chamfer distance respectively. Baseline is the PolyVectorization and ours is our proposed method with the finetuned GPT-3.5 models.

Fold	RPD		CD	
	Baseline	Ours	Baseline	Ours
1	30.059	14.295	2.417	1.771
2	29.656	14.807	2.288	1.504
3	29.737	14.904	2.439	1.689
4	30.040	14.952	2.438	1.949
5	29.953	14.435	2.367	1.653
Average	29.889	**14.678**	2.390	**1.713**

For each character within the testing dataset, rasterized pixel distance and Chamfer distance were calculated using the generated skeleton image and its corresponding ground truth skeleton image, and the average values for each testing group were recorded.

As a baseline, two metrics were also computed for font skeletons generated by PolyVectorization [7], one of the existing skeleton generation methods we mentioned in Section 2.2. Chamfer distance was computed with the points extracted from polyline format of PolyVectorization.

The results show that our proposed method outperforms the baseline method in both metrics.

Additionally, to compare the model's performance on Chinese characters and other characters, we calculated the Chamfer distance for both groups. We chose Chamfer distance over rasterized pixel distance to avoid bias towards simpler characters with more white pixels. The results show that our fine-tuned model generates Chinese characters more accurately, with an average Chamfer distance of 1.528, compared to 2.781 for other characters. This difference indicates that our model is more effective at generating Chinese characters than other characters.

5.3 Qualitative Evaluation

We conducted visual analysis to qualitatively compare the generated skeletons with its corresponding outlines and ground truth skeletons. We found that the majority of generated skeletons closely resemble the ground truth.

Comparison to PolyVectorization In order to compare generated results between our model and PolyVectorizaiton, we illustrate both results in Fig. 6. The generated skeletons were depicted in blue for both our method and PolyVectorization, red for the ground truth, and grey for the overlapped parts, making it easier to see the differences between the generated skeletons and the ground truth.

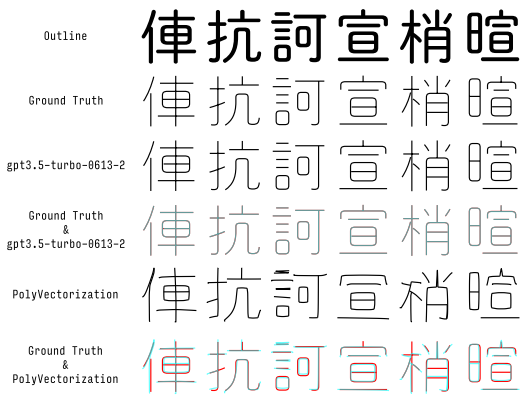

Fig. 6: Generated font skeletons for visual analysis (red is ground truth, blue is generated skeleton, grey is the overlapped part). gpt3.5-turbo-06123-2 is the fine-tuned model for the fold 2.

From Fig. 6, we can see that the skeletons generated by our model were almost identical to the ground truth. The lines and curves generated by our model not only corresponded accurately to the positions of the ground truth but were also very neat and smooth. In contrast, the skeletons generated by

PolyVectorization, while similar to the ground truth, were characterized by rough lines and the presence of unnecessary lines and noise in some components. These visual comparisons demonstrate that our model produces better font skeletons than PolyVectorization.

Failed Generation Patterns There are also some failed generation patterns (see Fig. 7) in generated skeletons by our model. We found that most of the failed patterns in Chinese characters are within very complex characters, some of which include rarer radicals. Moreover, contrary to our expectations, many Japanese kana, Latin, and symbol characters with very simple structures also failed to be generated. Based on our visual analysis, we speculate that this may be due to the scarcity of similar components, such as the rarer radicals in complex Chinese characters, as well as Japanese kana, Latin, and symbol characters in the training data. This scarcity could result in insufficient learning by the model, leading to the generation of failed patterns.

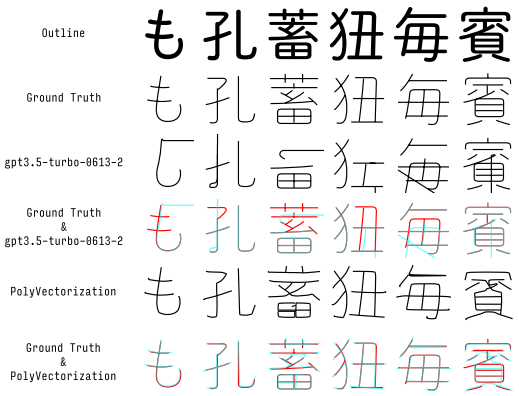

Fig. 7: Failed generated patterns in generated font skeleton (red is ground truth, blue is generated skeleton, grey is the overlapped part)

Radicals in Chinese Characters In order to further investigate failed generation patterns, we visually analyzed the Chinese characters focusing on radicals, the shared constituents across different characters. We performed the analysis on radicals with different complexities (number of strokes) as shown in Fig. 8.

As the number of strokes in a radical increases, we observed a higher occurrence of failed patterns in the generated skeletons. Specifically, radicals with more than 10 strokes were more likely to exhibit incorrect stroke paths, redundant strokes, or missing strokes in generated skeletons.

We also observed that skeletons with relatively simple radicals were consistently well-generated across different Chinese characters, and their accuracy was

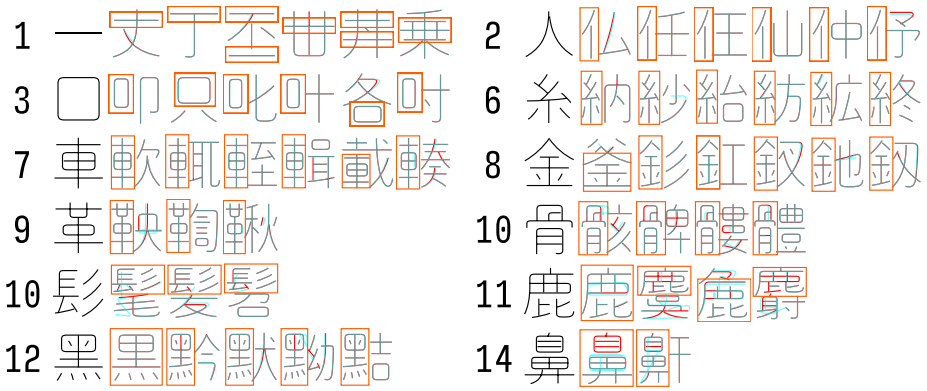

Fig. 8: Radicals in generated font skeleton of Chinese characters (red is ground truth, blue is generated skeleton, grey is the overlapped part, orange indicates radicals, numbers represent number of stroke of the radical)

not compromised by changes in the radicals' positions within the characters (see 1, 3, 8 in Fig. 8). This suggests that our model effectively learned the correspondence between outline information and skeleton paths for simple radicals, maintaining generation accuracy despite positional variations within the Chinese characters.

For complex radicals with over 10 strokes, the limited examples in the training data likely impeded the model's ability to fully learn the accurate correspondences between outlines and skeletons, leading to failed generation results. This may also explain the unsatisfactory generation results for some simple characters, such as Japanese kana, Latin, and symbol characters. Both results suggest that insufficient representation in the training data is affecting the model's performance.

Textual Representations of Chinese Characters In addition to graphical representations, we also compared textual representations of the generated skeletons with those of the ground truth, focusing on the stroke order and the direction of each stroke. We selected the top 100 Chinese characters based on Chamfer distance. We found that only 34 of the 100 generated skeletons completely follow the stroke order of the ground truth.

This result demonstrates that even when the visual similarity between the generated skeletons and the ground truth is high, discrepancies can still exist in the textual representations including disordered strokes, misdirected strokes, and split or redundant strokes (see Fig. 9).

Fig. 9 displays the generated results for characters excluded from the training data, while Fig. 10 shows results for characters included in the training data. Notably, similar discrepancies occur in both cases, regardless of whether the characters were used to train the model.

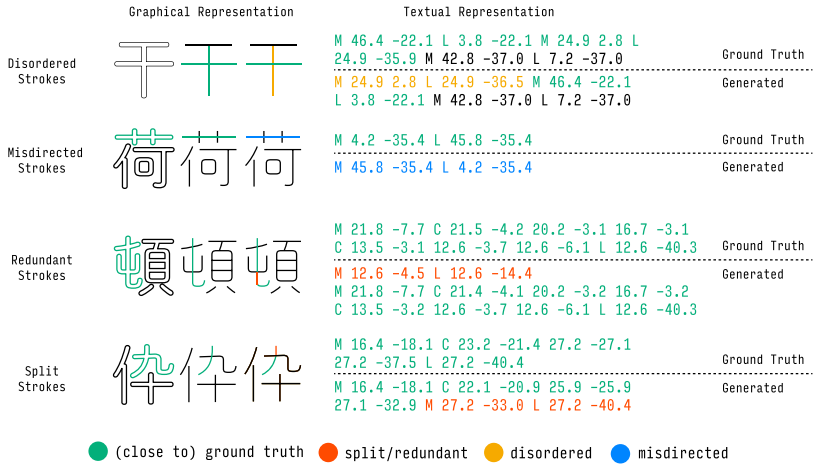

Fig. 9: Discrepancies in the textual representations of generated skeletons for characters not included in the training data of fold 2 (color coding: green - strokes close to ground truth; red - split or redundant strokes; orange - disordered strokes; blue - misdirected stokes)

These discrepancies highlight that our model can successfully replicate graphical representations without necessarily reproducing the exact stroke order and direction present in the ground truth data. Since variations in stroke order and direction do not significantly affect the usability of skeleton data for most applications, this finding presents an opportunity for data augmentation. Future research could explore intentionally randomizing stroke order and direction in the training data, potentially improving the model's robustness and generalization capabilities.

5.4 Limitation

One major limitation of this work is the dependency on a high-quality manually annotated dataset of paired font outlines and skeletons for training the language model. Although the size of the data is modest compared to other vision tasks, it is still laborious work given the complexity of the annotation task. While the 8,213 character pairs used in the experiments cover major characters in Japanese, it is still limited in scope to a single font style. Extending this approach to generate skeletons for a wider variety of fonts would require significantly more training data across different typeface designs.

Some examples of the generated skeletons for the additional fonts M Plus Rounded 1c, Noto Sans, and Noto Serif, as mentioned in Section 3, are shown in Fig. 11. As observed, the majority of these generated skeletons exhibited failed patterns. However, we found that a few of the generated skeletons for M Plus Rounded 1c, a rounded sans-serif font, achieved relatively ideal shapes

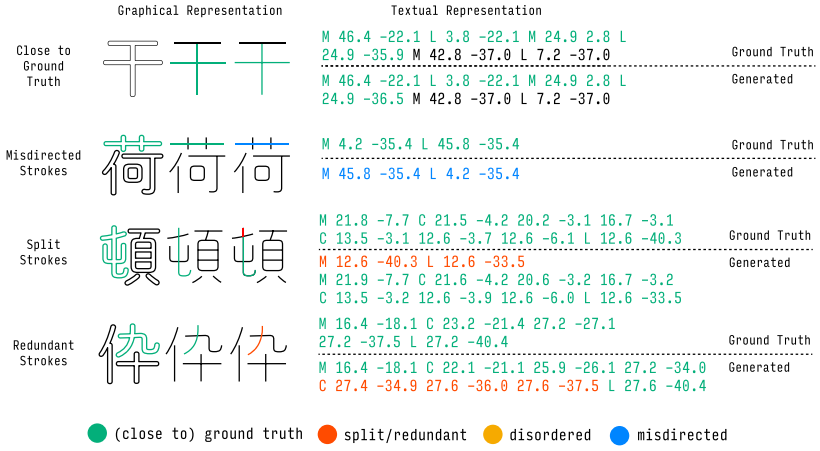

Fig. 10: Discrepancies in the textual representations of generated skeletons for characters included in the training data of fold 3 (color coding: green - strokes close to ground truth; red - split or redundant strokes; orange - disordered strokes; blue - misdirected stokes)

that appeared closer to the intended skeletons. This could be attributed to the similarity between the outlines of M Plus Rounded 1c and the font used for training data, DNP Shuei MGothic Std L. Although preliminary, this observation suggests that the proposed approach may have some degree of robustness in generating skeletons for fonts with outlines comparable to those in the training data.

Additionally, the fine-tuning process for commercial models such as GPT-3.5 offers limited ability to customize hyperparameters and architectures beyond a few basic settings such as epochs and learning rates. More flexibility in fine-tuning introduced by open source models could potentially improve the quality of the generated skeletons.

Finally, The evaluation metrics employed, while providing a quantitative benchmark, may not fully capture the nuances and qualitative aspects that human designers look for in high-quality font skeletons. More work is needed to develop robust evaluation methods tailored specifically for this task.

6 Conclusion and Future Work

This research presents a pioneering effort to leverage the capabilities of LLMs for the novel task of automatic font skeleton generation from outline data. By fine-tuning the GPT-3.5 model on a dataset of paired outlines and skeletons, the study demonstrates the feasibility of using LLMs to produce useful font skeleton resources.

Both quantitative metrics and qualitative visual analysis showed that the generated skeletons closely matched the ground truth data for the majority of

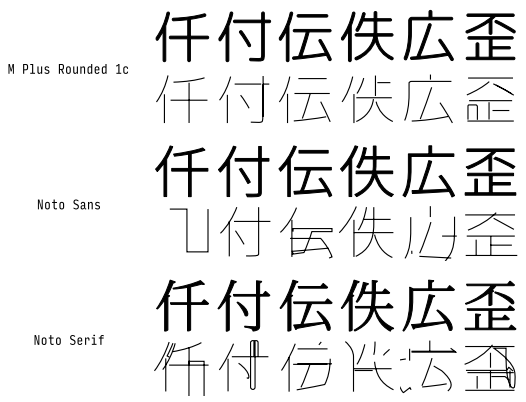

Fig. 11: Generated skeletons of additional fonts

characters in the test set, outperforming the existing algorithm-based method, PolyVectorization. This highlights the potential of data-driven approaches powered by LLMs for this task. And our model also demonstrates the ability to map skeleton path information that partially corresponds, even in the presence of variations in the ground truth sequence.

However, the study also revealed some key limitations and areas for improvement. Generation failures for certain subsets of complex and simple characters suggest potential biases or gaps in the training data. Scaling up the approach will also require significantly larger and more diverse datasets spanning multiple font styles and glyph sets.

Looking ahead, future work should explore techniques to improve customization during the fine-tuning process, develop better evaluation metrics tuned for font skeletons, and incorporate domain-specific knowledge about stroke structures and character components.

Overall, this application of LLMs has opened up a new direction for automating font skeleton generation, with implications for streamlining typeface design workflows and expanding the availability of high-quality font resources. Continued research building on these initial findings can help unlock the full potential of data-driven methods for this challenge in research and practice of typography.

Acknowledgement. The use of the outline and skeleton data of DNP Shuei Mgothic Std in this research is approved by Dai Nippon Printing. This research is supported by Google Award for Inclusion Research, JSPS KAKENHI Grant Numbers JP21K21311 and JP22K18137, JST CRONOS Japan Grant Number JPMJCS24K4, and JST SPRING Japan Grant Number JPMJSP2106.

References

1. Alexandr, N., Irina, O., Tatyana, K., Inessa, K., Arina, P.: Fine-tuning gpt-3 for russian text summarization. In: Data Science and Intelligent Systems: Proceedings of 5th Computational Methods in Systems and Software 2021, Vol. 2. pp. 748–757. Springer (2021)
2. Aoki, H., Aizawa, K.: Svg vector font generation for chinese characters with transformer. In: 2022 IEEE International Conference on Image Processing (ICIP). pp. 646–650. IEEE (2022)
3. Apel, U., Quint, J.: Data on the construction of kanji and various approaches for their usage
4. Apel, U., Quint, J.: Building a graphetic dictionary for japanese kanji-character look-up based on brush strokes or stroke groups, and the display of kanji as path data. In: Proceedings of the Workshop on Enhancing and Using Electronic Dictionaries. pp. 36–39 (2004)
5. Beier, S., Bernard, J.B., Castet, E.: Numeral legibility and visual complexity. In: DRS Design Research Society, 2018: Limerick (2018)
6. Berio, D., Leymarie, F.F., Asente, P., Echevarria, J.: Strokestyles: Stroke-based segmentation and stylization of fonts. ACM Transactions on Graphics (TOG) **41**(3), 1–21 (2022)
7. Bessmeltsev, M., Solomon, J.: Vectorization of line drawings via polyvector fields (2018)
8. Brown, T.B., Mann, B., Ryder, N., et al.: Language models are few-shot learners (2020)
9. Chen, M., Tworek, J., Jun, H., et al.: Evaluating large language models trained on code (2021)
10. Cox, C., Coueignoux, P., Blesser, B., Eden, M.: Skeletons: A link between theoretical and physical letter descriptions. Pattern Recognition **15**(1), 11–22 (1982). https://doi.org/https://doi.org/10.1016/0031-3203(82)90056-5, https://www.sciencedirect.com/science/article/pii/0031320382900565
11. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR **abs/1810.04805** (2018), http://arxiv.org/abs/1810.04805
12. Garg, S., Ramakrishnan, G.: Bae: Bert-based adversarial examples for text classification. arXiv preprint arXiv:2004.01970 (2020)
13. Herz, J., Hersch, R.D., Gonczarowski, J.: Coherent processing of character skeletal forms. Computers & Graphics **21**(6), 727–736 (1997). https://doi.org/https://doi.org/10.1016/S0097-8493(97)00050-2, https://www.sciencedirect.com/science/article/pii/S0097849397000502, graphics in Electronic Printing and Publishing
14. Jakubiak, E.J., Perry, R.N., Frisken, S.F.: An improved representation for stroke-based fonts. In: ACM SIGGRAPH 2006 Sketches. p. 137–es. SIGGRAPH '06, Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1179849.1180020, https://doi.org/10.1145/1179849.1180020
15. Jia, Y., Zhang, Y., Weiss, R., Wang, Q., et al.: Transfer learning from speaker verification to multispeaker text-to-speech synthesis. Advances in neural information processing systems **31** (2018)
16. Karow, P.: Digital typefaces: description and formats. Springer Science & Business Media (1994)
17. Knuth, D.E.: The METAFONT book. Addison-Wesley Longman Publishing Co., Inc. (1989)

18. Lupton, E.: Thinking with type: A critical guide for designers, writers, editors, & students. Chronicle Books (2014)
19. Mathey, Y., et al.: Prototypo. https://mathey.notion.site/Prototypo-22a55139f0324a7eb82a67bbceaa3a58 (2014)
20. OpenAI: Fine-tuning. https://platform.openai.com/docs/guides/fine-tuning (2020)
21. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al.: Training language models to follow instructions with human feedback. Advances in neural information processing systems **35**, 27730–27744 (2022)
22. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., et al.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
23. Qu, C., Yang, L., Qiu, M., Croft, W.B., Zhang, Y., Iyyer, M.: Bert with history answer embedding for conversational question answering. In: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval. pp. 1133–1136 (2019)
24. Quint, J., Apel, U.: Does learning how to read japanese have to be so difficult: and can the web help? In: Special interest tracks and posters of the 14th international conference on World Wide Web. pp. 1152–1153 (2005)
25. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al.: Improving language understanding by generative pre-training (2018)
26. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
27. Reyes, A.K., Caicedo, J.C., Camargo, J.E.: Fine-tuning deep convolutional networks for plant recognition. CLEF (Working Notes) **1391**, 467–475 (2015)
28. Samara, T.: Letterforms: Typeface Design from Past to Future. Rockport Publishers (2018)
29. Scheichelbauer, R.E.: Lttr/ink 1.0 released. https://glyphsapp.com/news/lttr-ink-1-0-released (2021)
30. Souza, F., Nogueira, R., Lotufo, R.: Portuguese named entity recognition using bert-crf. arXiv preprint arXiv:1909.10649 (2019)
31. 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., Lample, G.: Llama: Open and efficient foundation language models (2023)
32. Zong, M., Krishnamachari, B.: Solving math word problems concerning systems of equations with gpt-3. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 15972–15979 (2023)