UNIVERSAL BRAILLE-TO-TEXT TRANSLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

For visually impaired readers, text is transcribed into braille under a transcription code, a set of rules specific to a language and a standard, and every code writes into the same 256 Unicode braille codepoints. Given the code, transcription is deterministic, but its inverse is not: several texts can produce the same cells, and the cells do not mark which of hundreds of codes produced them. This inverse, braille-to-text translation, is how transcriptions are proofread and how documents that survive only in braille are recovered. Existing systems require the code, most are built one code at a time, and even the widely used rule-based engine, LibLouis, fails to invert its own output when given the code. In response, we present a system that infers the code and reads braille across 156 codes in 85 languages. We fine-tune an LLM on pairs generated by the deterministic text-to-braille transcriber, and its candidate readings are verified by re-transcription. Our system achieves 0.39% per-cell error, against 11.4% for LibLouis given the correct code. We also analyze the remaining error by whether re-transcription can detect it, and find that it concentrates in a few code families, each for a different cause. On an English Bible, with the code given, 80% of readings that reproduce the input cells read at 0.29% character error, against 2.86% for the rest. We open-source our implementation, corpus, and checkpoints: `https://github.com/anonytoucan/ubt`.

1 INTRODUCTION

Braille, the tactile writing system of visually impaired readers, is written in cells of six dots (sometimes eight), and every braille code, a set of transcription rules specific to a language and a standard, is encoded in the same 256 Unicode braille codepoints (U+2800–U+28FF). Once a code is selected, transcribing text to braille is deterministic; the inverse is not. This inverse, back-translation, is how documents written in braille are recovered as text and how transcriptions are proofread. Across codes, the same cells map to different text, and a cell sequence carries no marker of its code, so inversion has to recover both the text and the code. Even within a code, text-to-braille transcription is many-to-one, because the cells are far fewer than the characters and words they must write: in codes that use contractions the same cell can stand for a letter, a letter group, or a whole word, and several distinct characters are written with the same cells. The cells alone do not choose among these readings: the code must be given by the user or inferred, and among the texts it allows, the choice has to come from knowledge of the language.

Conventional back-translators are rule-based (e.g., Blenkhorn, 1995) (Figure 1a); LibLouis (Egli, 2009) is widely used. Each code is implemented as a rule table, and the user has to specify the code of each segment. Data-driven back-translators each handle a single code (Jiang et al., 2002; Seto et al., 2023; Wu et al., 2024), and a recent system (Huang et al., 2025) built on a large language model (LLM) handles Chinese and English braille mixed with mathematics, with the language named in the prompt. To our knowledge, no system infers the code from the cells, although documents commonly mix codes (e.g., Nemeth mathematics inside prose, Latin inside a non-Latin base). Existing tools, taking the code as given, cannot handle such documents on their own.

In response, we propose a universal braille-to-text translation (UBT) system: a single model that infers the code of each line and translates braille back to text across 156 codes in 85 languages (Figure 1b). We use the determinism of text-to-braille transcription in two ways. First, we generate exact (text, cells) pairs with the transcriber from a monolingual corpus, so a language model can be fine-tuned without human-transcribed documents. Second, we check a candidate reading by re-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

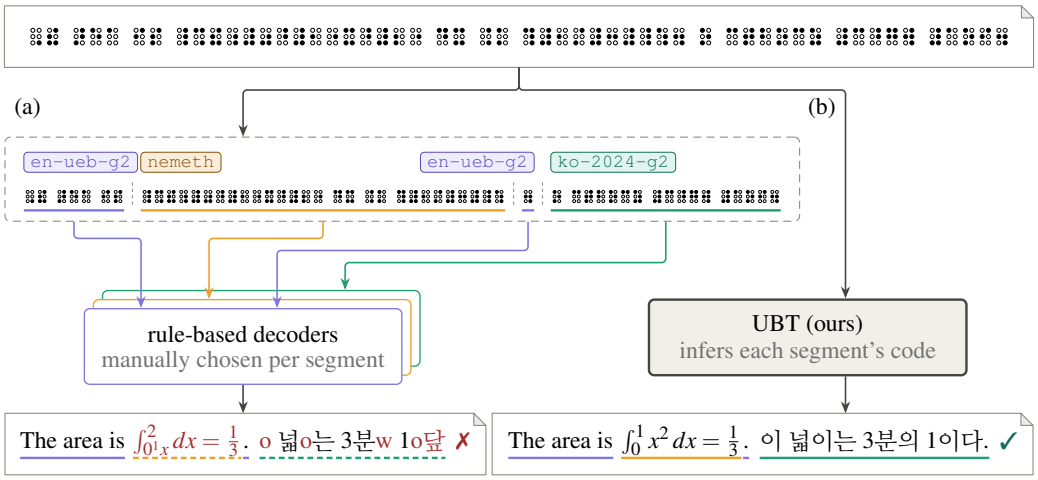


Figure 1: **Braille-to-text translation.** The input is braille whose segments may use different, unmarked codes. (a) A rule-based decoder needs each segment's code and applies one rule table per segment. (b) Our system receives only the cells and jointly infers each segment's code and text.

transcribing it under its predicted code, and call it consistent if this reproduces the input cells. At inference we sample several candidates and return the most probable consistent one, or, if none is consistent, the closest one. The system is implemented by fine-tuning Qwen2.5-7B (Yang et al., 2024) with LoRA (Hu et al., 2022).

We evaluate UBT on held-out synthetic documents and on real-world braille. On the synthetic documents, without being given the code, it has 0.36% CER, against 11.76% for LibLouis given the correct code and 18.96% for GPT-5.5 (OpenAI, 2026), a general-purpose LLM. Since CER is not comparable across scripts, we also report an error rate on braille cells (bCER), on which the margins are similar. It identifies the code of 90.3% of the segments, and its outputs are the closest to the references in meaning, as measured by a multilingual embedding model (Chen et al., 2024) and an LLM judge. We further analyze the selection at inference, the training stages and the backbone, documents that mix codes, the kinds of error that re-transcription can and cannot detect, and the error of each code. Most of the remaining errors reproduce the input cells and thus cannot be detected without the reference, and they concentrate in a few code families, each for a different cause, such as homophones in Chinese and codes confused across languages. On four real-world braille corpora, our system remains accurate when the braille follows our codes, and when the braille follows conventions outside our codes, fine-tuning only on a publicly released in-domain training set makes it more accurate than a domain-specific system trained on more data. We release the corpus generators, generated corpora, evaluation adapters, checkpoints, and a braille edition of this paper.

2 Related Work

Rule-based braille translation. Braille translation software has been rule-based since its early finite-state formulations (Millen, 1970; Sullivan, 1975). As in Slaby (1980) and the back-translator of Blenkhorn (1995), one language-independent engine applies a hand-written table of rules for each code. LibLouis (Egli, 2009) follows this design and is widely used, including in screen readers. We use its transcription tables as our transcriber. Its back-translation, our rule-based baseline, reuses most of the transcription rules in reverse. Where several texts share the same cells, it picks one by fixed rules, such as the order of definitions, and so does not always recover the original text.

Learned back-translation. Statistical and neural back-translators instead use a language model to choose among the readings the cells allow. They have usually targeted a single language, most often Chinese, whose braille writes syllables that many characters share and is therefore hard to invert. Jiang et al. (2002) followed rule-based conversion of braille into pinyin, the standard romanization of Mandarin, with an n-gram language model, and Huang et al. (2024) and Wu et al. (2024) later

fine-tuned pretrained language models. Wu et al. (2024) trained on pairs produced by a transcriber, as we do, but did not use it at inference. For Japanese, Seto et al. (2023) trained a neural model to convert the syllabic kana that braille writes into kanji, the Chinese characters used in Japanese. The closest prior system, BrailleLLM (Huang et al., 2025), is an LLM fine-tuned on Chinese and English braille, each mixed with mathematics, with the language named in the instruction. Even recent LLMs still perform poorly on braille without such fine-tuning (Abdullah, 2026; Zhang et al., 2026). To our knowledge, ours is the first learned back-translator that covers a wide range of languages and codes, infers the code from the cells, and checks its readings against the transcriber at inference.

Translation and inversion. Braille back-translation and our approach to it resemble translation and inversion work in the natural language processing (NLP) literature in two respects: training on synthetic pairs, and recovering text from a many-to-one encoding. Machine translation synthesizes pairs by running a reverse model over monolingual text, which it calls back-translation (Sennrich et al., 2016; Edunov et al., 2018), and repeats this as both directions improve (Hoang et al., 2018). Round-trip reconstruction has also served as a training signal (He et al., 2016). For us, the transcriber plays the role of the reverse model. Like braille back-translation, pinyin-to-character conversion (Chen & Lee, 2000), kana-to-kanji conversion (Tokunaga et al., 2011), and diacritic restoration (Zitouni et al., 2006) choose among texts that share one encoded form, using knowledge of the language, but their encoding is fixed. The braille code, in contrast, must be inferred from the cells. Optical character recognition (OCR) post-correction (Nguyen et al., 2021) and noisy-channel decoding (Yee et al., 2019) likewise combine a language prior with a model of the channel, but they learn the channel. The braille channel, in contrast, is a deterministic program, the transcriber, which yields exact training pairs and an exact round-trip check.

3 System

3.1 Problem Formulation

Notation. Braille B is mostly written in cells of six dots, such as "⠛" and "⣿", and some codes use eight, so every code writes into the same 256 Unicode braille characters (U+2800–U+28FF). A transcription code c is a set of rules that writes the text of a language under a given standard in braille cells, and its transcriber f_c is a deterministic program that maps a text to its cells. A document consists of m segments separated by line breaks, so the segment boundaries are visible in the input. Segment j has text T_j and code c_j, and writing $|$ for a line break, the document has text $T = T_1 \mid \cdots \mid T_m$, codes $\theta = (c_1, \ldots, c_m)$ and braille

$$B = f_\theta(T) = B_1 \mid \cdots \mid B_m, \qquad B_j = f_{c_j}(T_j). \tag{1}$$

Our task is to recover the codes $\hat{\theta}$ and the text $\hat{T}$ of every segment from B alone, and we write the output as $\hat{Y} = (\hat{\theta}, \hat{T})$. We call $\hat{Y}$ *consistent* with the input if its re-transcription reproduces the input, $\hat{B} = f_{\hat{\theta}}(\hat{T}) = B$. Note that since the transcriber is deterministic, consistency can be checked without the ground-truth text, which could be helpful when only the braille is available.

Challenges. Back-translation is nontrivial and challenging for two reasons. First, across codes, different codes can emit the same cells for different texts or for the same text. For example, the cell "⠛" stands for the letter "b" in uncontracted codes (e.g., `en-ueb-g1`, `de-g0`), for the word "but" in contracted English (`en-ueb-g2`) and for a syllable-final "ㅂ" in Korean (`ko-g1`). For the same text, the 2006 and 2024 Korean codes emit identical cells in most digit-free sentences. Second, even within a code, the transcriber is many-to-one: contractions, capital and number indicators, and characters the code does not distinguish let several texts share one cell string. The cells cannot tell these texts apart, so only the language prior can choose among them.

3.2 Model

Method. We use a language model that takes the input braille and emits the output one segment per line: a code prefix with the inferred code, followed by the text, $\langle \hat{c}_j \rangle \hat{T}_j$ (Figure 2). The prompt is the input braille under a <|braille|> tag, followed by a <|text|> tag; in training, the completion is $\langle c_1 \rangle T_1 \mid \cdots \mid \langle c_m \rangle T_m$. Unlike the language token of multilingual translation (Johnson et al., 2017), which is given to select the output language, the code prefix is predicted by the model.

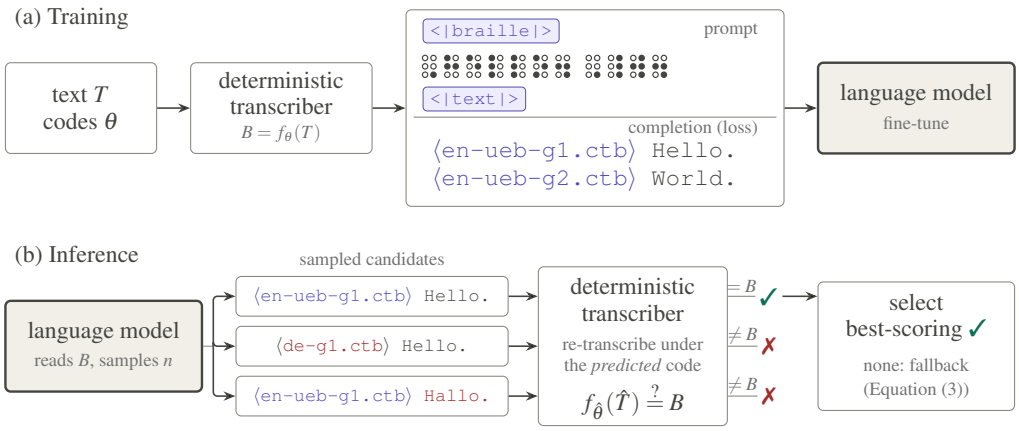


Figure 2: **Training and inference with the deterministic transcriber.** In training, the transcriber generates the (text, cells) pairs. At inference, the same transcriber is used to re-transcribe the sampled candidates, evaluate their consistency, and select the best candidate.

LLM prior. Since the transcriber is deterministic, the posterior over outputs given the input braille is proportional to the prior over outputs restricted to the consistent ones, so the model has to supply a prior over text. We therefore fine-tune a pretrained language model, Qwen2.5-7B (base) (Yang et al., 2024), with a LoRA adapter (Hu et al., 2022) on the training documents, with the next-token loss on the completion only. The adapter has rank 64, α=128 and dropout 0.05 on all linear layers. Each braille cell is treated as a single token: the 256 braille codepoints are added to the vocabulary, and each new embedding is initialized to the mean embedding of the byte-level tokens it replaces.

Training data. Training documents are generated by transcribing source texts from Wikipedia and the dev set of FLORES-200 (Costa-Jussà et al., 2022), a set of Wikimedia sentences translated into 200 languages, with LibLouis 3.38 (Egli, 2009), which we use with some modifications, including a Korean table repaired against the 2024 standard, a separate backend that transcribes mathematics from LaTeX under the Nemeth code, and the merging of tables that produce identical cells. We train on 156 codes spanning 85 languages and hold out eight further codes, six of which serve as zero-shot codes. Codes are sampled with weights determined by the status of their standard and their speaker population. Furthermore, to simulate real-world braille, we construct four types of documents. (i) Single-code documents are transcribed under one code. (ii) Intra-code switching documents insert Latin terms, digits and dates inside one code, taking naturally occurring Latin from the source corpora where possible. (iii) Inter-code mixtures combine two to four codes, chosen in three ways: a matrix language with high-resource embedded languages, uniformly at random, or among codes of the same language or script that can emit the same cells, such as `ko-2024`/`ko-2006`. (iv) As a special case, mathematics documents are LaTeX expressions in Nemeth braille, half standalone and half embedded in Unified English Braille (UEB) (International Council on English Braille, 2013) prose between Nemeth switch indicators. Of the 941,775 training documents, 66.8% are single-code, 12.7% intra-code switching, 17.8% inter-code mixtures and 2.7% mathematics.

Optimization. We train our model in three stages. First, we train for one epoch on our main training data with AdamW (Loshchilov & Hutter, 2019), a learning rate of 1.5×10^{-4} ($\times 10$ higher for the new cell-token embeddings), a cosine schedule with 2% warmup, no weight decay, gradient clipping at 1.0 and a batch size of 128 documents of at most 4,096 tokens. Second, for the codes with the highest error after the first epoch (the Japanese and Chinese codes and 15 contracted codes), we run a continuation round of one epoch on new documents, with replay of the original corpus, at a learning rate of 5×10^{-5}. Finally, to further improve the performance, we add an extra LoRA adapter of the same rank and train it with Group Relative Policy Optimization (GRPO) (Shao et al., 2024) with reward given as

$$r(\hat{Y}) = \begin{cases} -0.5 & \text{if unparsable,} \\ \max\big(0,\, 1 - \mathrm{CER}(\hat{T}, T)\big) + 0.5\,[\hat{B} = B] + 0.25\,[\hat{T} = T] + 0.25\,[\hat{\theta} \simeq \theta] & \text{otherwise,} \end{cases} \tag{2}$$

where $[\cdot]$ is 1 if the condition holds and 0 otherwise, and $\hat{\theta} \simeq \theta$ holds if every predicted code writes the reference text with the input cells. Training runs for 400 steps of 24 prompts, with 8 samples per prompt, at a learning rate of 10^{-5}.

3.3 Inference

Best-of-n selection. At inference we sample n candidate outputs $\hat{Y}_1, \ldots, \hat{Y}_n$ and compute the re-transcription $\hat{B}_i$ of each, re-transcribing every segment under the code the candidate predicted for it. This is best-of-n selection (Cobbe et al., 2021) with the transcriber as the verifier. Among the candidates consistent with the input, we return the one with the highest model probability q; if none is consistent, we fall back to the candidate that scores best considering both the log-probability and the cell-level edit distance d_{cell} to the input, weighted by λ:

$$\hat{Y} = \begin{cases} \arg\max_{i:\,\hat{B}_i = B} \log q(\hat{Y}_i \mid B) & \text{if some } \hat{B}_i = B, \\ \arg\max_i \left[\log q(\hat{Y}_i \mid B) - \lambda\, d_{\text{cell}}(\hat{B}_i, B) \right] & \text{otherwise.} \end{cases} \tag{3}$$

We use n=20, the greedy decode and 19 samples at temperature τ=0.8 and top-p 0.95, and λ=2, chosen in preliminary experiments. Full implementation details are in Appendix A.

Justification. A natural alternative to our selection is constrained decoding, which restricts generation token by token to outputs whose prefixes remain valid, for example to a formal language (Willard & Louf, 2023; Geng et al., 2023; Beurer-Kellner et al., 2024) or to the prefixes a parser accepts (Scholak et al., 2021). This requires a test on partial outputs, which transcription does not provide. Transcription is not prefix-stable, since the cells for a character can depend on the characters that follow (in UEB, "th" is "⠹" but "the" is "⠮"). Hence, a partial output cannot be checked against a partial input. Checking complete candidates instead uses the transcriber unchanged. It also selects among samples drawn from the model's own distribution, whereas masking invalid tokens at each step distorts that distribution (Park et al., 2024). Finally, this soft selection avoids the engineering burden of constraining the decoder and is easy to handle and to adapt, as we show in the real-world braille experiments.

4 Evaluation

4.1 Setup

Evaluation data. We evaluate on a held-out test set of 20,071 sentence-level documents over the 156 codes, built from the devtest subset of FLORES-200 and a held-out Wikipedia shard with the same generator as the training data. We also evaluate on four corpora of human-produced braille: 3,000 verses of an English Bible (Bibles for the Blind and Visually Handicapped International, 2024), the NIKL Korean text–braille parallel corpus (National Institute of Korean Language, 2024), and two Chinese sets, the BrailleLLM test set (Huang et al., 2025), corrected by hand after machine conversion, and the Vision-Braille release (Wu et al., 2024), whose braille marks tones. More details on the corpora are in Appendix B.

Metrics. Following prior work (Huang et al., 2025), we report the character error rate (CER) as our primary metric. A cell encodes a different number of characters in each script, so CER is not comparable across scripts. We therefore also propose and report the braille-cell error rate (bCER), which counts errors per cell. It aligns the reference text to its cells through the transcriber's position map, projects each character edit onto the cells it covers, and divides by the number of cells. For Chinese, Traditional characters are folded into Simplified characters before scoring, since the Chinese codes write both with the same cells. We also measure code accuracy (Code Acc.) in two ways. The strict form credits only the code that produced the input, and the confusability-aware form (conf.) also credits another code that transcribes the predicted text into the same input cells. To see whether the errors change the meaning, we also report two semantic scores. The first is the cosine similarity (sim.) between the dense embeddings of the output and the reference from a pretrained multilingual embedding model, bge-m3 (Chen et al., 2024), scaled to 0 to 100. The second is the score of an LLM judge, Qwen2.5-72B-Instruct-AWQ (Yang et al., 2024), instructed to rate from 1 to 5 how faithfully the output reproduces the meaning, wording, names, numbers and symbols of the reference. The full details of the metric computation, including the judge prompt, are in Appendix C.

Table 1: **Results on the main test set.** †Given the ground-truth code.

	CER (%)	bCER (%)	Code Acc. (%)		sim.	judge
			strict	conf.		
LibLouis backward†	11.76 ±0.34	11.36 ±0.32	–	–	89.6 ±0.2	3.99 ±0.02
GPT-5.5, few-shot†	18.96 ±2.74	19.03 ±2.55	–	–	89.4 ±1.3	4.14 ±0.10
UBT (ours)	0.36 ±0.03	0.39 ±0.03	90.3 ±0.4	97.6 ±0.2	99.4 ±0.0	4.90 ±0.01
w/ oracle code†	**0.27 ±0.01**	**0.31 ±0.02**	–	–	**99.5 ±0.0**	**4.91 ±0.01**
greedy decoding	0.54 ±0.04	0.57 ±0.04	89.5 ±0.4	96.7 ±0.2	99.1 ±0.0	4.83 ±0.01
w/o GRPO (stage 2)	0.42 ±0.03	0.44 ±0.03	90.2 ±0.4	97.7 ±0.2	99.3 ±0.0	4.87 ±0.01
base training only (stage 1)	0.53 ±0.03	0.61 ±0.04	**90.7 ±0.4**	98.2 ±0.2	99.2 ±0.0	4.83 ±0.01
ByT5 base (stage 1)	0.81 ±0.06	1.19 ±0.09	89.0 ±0.4	96.2 ±0.3	98.7 ±0.1	4.82 ±0.01
Gemma-2-9B base (stage 1)	0.57 ±0.03	0.60 ±0.03	**90.7 ±0.4**	**98.3 ±0.2**	99.2 ±0.0	4.82 ±0.01

4.2 Main Results

Baseline comparison. We compare our system with the back-translation function of LibLouis. Since LibLouis requires the code, we provide the ground-truth code of each segment. We also compare with GPT-5.5 (OpenAI, 2026), a general-purpose LLM prompted few-shot. Table 1 reports the results with 95% bootstrap confidence intervals. With the code of each segment inferred, our system has 0.36% CER and 0.39% bCER, whereas LibLouis has 11.76% and 11.36%. Even with the correct code, the rule-based engine fails to invert its own output. GPT-5.5 has 18.96% CER, suggesting that general-purpose LLMs still struggle to back-translate braille, consistent with recent findings (Abdullah, 2026; Zhang et al., 2026). Figure 4 shows an example on which both baselines fail and UBT is correct. Our system identifies the code of 90.3% of the segments, and of 97.6% once the inherent ambiguity between codes that emit the same cells is taken into account. Given the oracle code, it is only slightly more accurate, with 0.27% CER and 0.31% bCER, so inferring the code costs little. Its outputs are also the closest to the references in meaning, with a similarity of 99.4 and a judge score of 4.90 out of 5, and the judge gives 97.8% of them a score of at least 4 (Figure 5).

Ablation study. We ablate the selection at inference, the training stages and the backbone. Without best-of-n, that is, with greedy decoding, CER increases by half. Best-of-n also raises code accuracy slightly. Note that reranking the same candidates by model probability alone, that is, without the consistency check and the cell edit distance of Equation (3), lowers CER only from 0.54% to 0.49%, compared with 0.36% for best-of-n, so the gain comes from the check against the input cells. Each training stage lowers the error. The continuation round (stage 2) improves the codes it targets, and GRPO (stage 3), whose reward also credits outputs that reproduce the input cells, lowers CER further while leaving code accuracy unchanged, so it improves the text rather than the code. To compare backbones, we train Gemma-2-9B and the byte-level ByT5 with stage 1 only. Our default choice (Qwen2.5-7B) has the lowest CER, and Gemma-2-9B matches it in code accuracy. ByT5 has a much higher bCER than CER, since its errors concentrate in scripts whose characters take several cells each. The results of other decoders and of each training stage are in Appendix D.

Multi-code texts. A document can contain segments with different codes (inter-code), and a single code can switch notation within a document (intra-code). We report the evaluation results on such documents separately in Table 2, where k is the number of codes. CER increases only slightly as k grows and remains close to that on single-code documents, and the intra-code documents do not increase it. Although the share of documents on which UBT predicts all the codes correctly (Code) decreases with more codes, CER remains close to that given the oracle code.

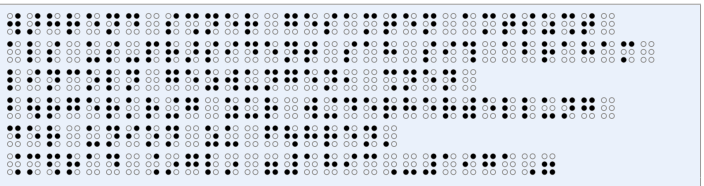
Table 2: **Multi-code text results.**

	UBT (ours)		w/ oracle
Documents	CER	Code	CER
single code	0.36	90.3	**0.27**
intra-code	**0.32**	**93.3**	0.34
inter-code, k=2	0.33	87.6	0.32
inter-code, k=3	0.38	81.5	0.35
inter-code, k=4	0.44	79.4	0.41

Table 3: **Back-translation error analysis results.** †Given the ground-truth code.

	inconsistent					consistent misreading					wrong code
	share	CER	bCER	sim.	judge	share	CER	bCER	sim.	judge	share
LibLouis backward†	31.8	30.39	26.83	74.3	2.52	20.8	11.25	13.53	89.3	3.92	–
GPT-5.5, few-shot†	41.3	42.63	43.07	74.9	2.95	8.8	8.85	9.01	97.4	**4.84**	–
UBT (ours)	6.3	2.50	2.63	95.5	4.04	7.2	1.70	1.99	97.5	4.52	8.6
w/ oracle code†	**5.8**	**2.19**	2.35	**95.8**	4.01	7.5	1.72	2.00	97.5	4.52	–
greedy decoding	14.4	2.62	2.70	95.5	4.04	**5.9**	**1.58**	**1.84**	**97.7**	4.58	**8.2**
w/o GRPO (stage 2)	10.1	2.28	**2.30**	95.7	**4.05**	6.8	1.65	1.92	**97.7**	4.55	8.5

Example 1
`de-g1 + de-g2-detailed`



Während seiner gesamten Amtszeit als US-Präsident sah sich Abraham Lincoln gezwungen, einen Bürgerkrieg zur Wiederherstellung der Union zu führen. Meinrad Inglin (1893–1971)

LibLouis ***inconsistent**, 20.2% CER*	während seiner gesamten amtszeit als us-präsident sah sich abraham lincoln gezwungen, einen bürgerkrieg zur wiederherstellung der union zu führen. M6ralld *g1* (allhatt9en–all97all[
GPT-5.5 ***consistent misreading**, 5.8% CER*	während seiner gesamten amtszeit als us-präsident sah sich abraham lincoln gezwungen, einen bürgerkrieg zur wiederherstellung der union zu führen. Meinrad Inglin (1893–1971)
UBT (ours), greedy ***inconsistent**, 0.6% CER*	Während seiner gesamten Amtszeit als US-Präsident sah sich Abraham Lincoln gezwungen, einen Bürgerkrieg zur Wiederherstellung der Union zu führen. Meinrad Inglingg (1893–1971)
UBT (ours) ***correct**, 0.0% CER*	Während seiner gesamten Amtszeit als US-Präsident sah sich Abraham Lincoln gezwungen, einen Bürgerkrieg zur Wiederherstellung der Union zu führen. Meinrad Inglin (1893–1971)

Figure 4: **Readings of an example input by each system.** Wrong characters are highlighted.

Error analysis. Back-translation systems can produce different kinds of error; see Figure 3. First, an error could be *inconsistent*, where the output does not reproduce the input cells, and only this kind is caught by re-transcription. Second, an error could be a *consistent misreading*, where the output reproduces the cells under the right code with a different text, which does not necessarily change the meaning. Third, an error could be a *wrong code*, where the output reproduces the cells using a different code. Table 3 reports the per-error-type evaluation results with the shares of documents. Best-of-*n* lowers the share of inconsistent outputs from 14.4% under greedy decoding to 6.3%, the documents on which no candidate is consistent, and raises the share of correct outputs from 71.5 to 77.9%. The two consistent kinds of error, consistent misreading and wrong code, rise slightly, from 14.1 to 15.8%. GPT-5.5 fails mostly with inconsistent outputs, whereas LibLouis also produces many consistent misreadings. For all three systems, consistent misreadings degrade the text less than inconsistent outputs, with higher similarity and judge scores. Figure 4 shows a multi-code document. LibLouis produces an inconsistent output and GPT-5.5 a consistent misreading. Greedy decoding of UBT errs in one character and is inconsistent, whereas best-of-*n* selects the correct candidate. Despite these errors, most outputs of UBT preserve the meaning (Figure 5). More examples, including a wrong code, are in Appendix F.

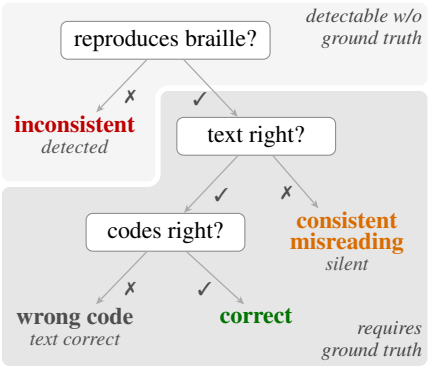


Figure 3: **Back-translation errors.**

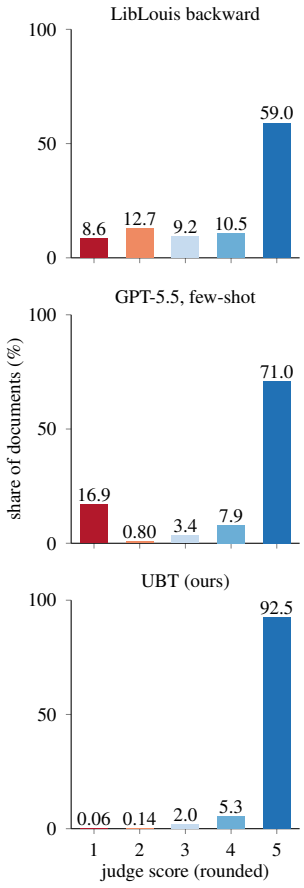


Figure 5: **LLM judge results.**

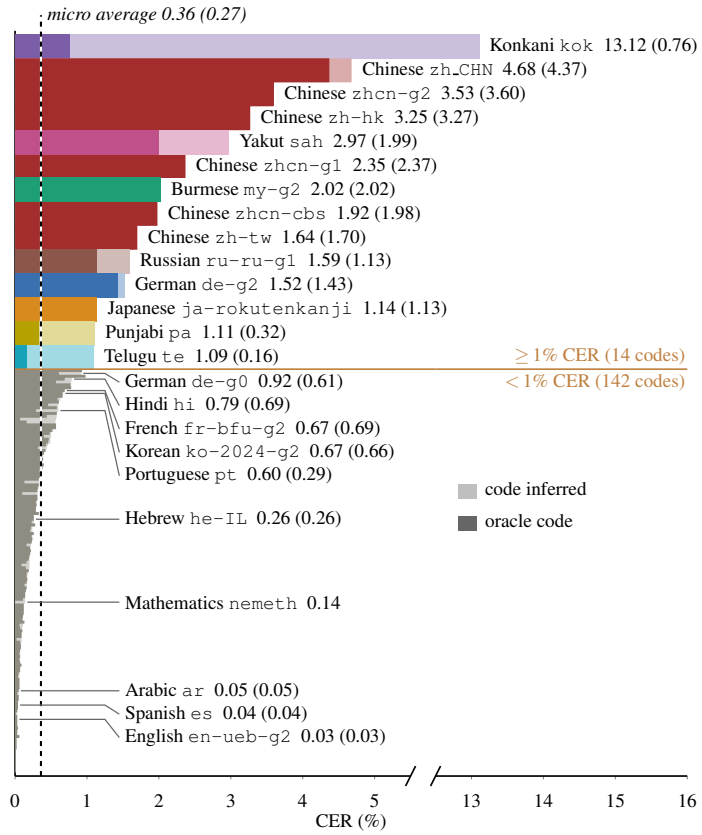


Figure 6: **Per-code CERs (sorted).** Codes with CER above 1% and ten others are labelled with their CER, with the CER given the oracle code in parentheses.

Per-code analysis. Figure 6 shows the CER of each of the 156 codes in sorted order, including mathematics. The error is concentrated in a few codes. Fourteen exceed 1% CER, six of them Chinese, and each family fails for a different reason. Chinese braille writes syllables, so most of the Chinese error consists of homophones, characters that share the syllable of the correct one and are thus written with the same cells, which makes them consistent misreadings. Konkani has the largest CER but a small one given the oracle code, since the model predicts the code of another language that emits the same cells. These outputs are wrong codes, and when the predicted code writes another script, the whole text is wrong. Punjabi and Telugu show the same pattern. In Japanese, the error is mostly rare kanji for which none of the candidates is correct. The contracted codes, such as German and Burmese grade 2, are mostly misread into inconsistent outputs. Re-transcription therefore detects the errors of the contracted codes but only part of those of Chinese and Konkani. The results for every family and code are in Appendices D and E.

Extra results. Refer to Appendices G, E and H for the results on codes held out from training, on mathematics and under dot-flip noise, respectively.

4.3 Results on Real Braille

Synthetic–real gap. Synthetic documents are generated by our transcriber, whereas real braille is produced by human transcribers and may deviate from our transcriber's output in three ways. First, braille does not encode some print distinctions. For example, the Bible writes "LORD" with a single capital sign rather than in full capitals. Such deviations are counted as CER errors even though the meaning is preserved, which motivates the semantic scores. Second, human transcribers may follow a convention, i.e., a transcription practice outside the standard codes, such as blank cells between

Table 4: **Results on real-world braille.** †Given the ground-truth code. ‡Not reproducible by our codes. §From Huang et al. (2025).

Corpus	Model	CER (%)	bCER (%)	sim.	judge
NIKL (Korean)	LibLouis backward†	54.38 ±0.13	31.20 ±0.06	74.0 ±0.1	1.41 ±0.00
	UBT (ours)	**0.85** ±0.02	**0.86** ±0.02	**99.7** ±0.0	**4.88** ±0.00
Bible (English)	LibLouis backward†	0.79 ±0.04	1.02 ±0.05	99.8 ±0.0	4.74 ±0.01
	GPT-5.5, few-shot†	**0.29** ±0.05	**0.33** ±0.06	**99.9** ±0.0	**4.98** ±0.01
	UBT (ours)	1.17 ±0.13	0.97 ±0.09	99.1 ±0.1	4.76 ±0.02
BrailleLLM test set (Chinese)‡	LibLouis backward†	35.30 ±0.69	31.24 ±0.59	71.8 ±0.6	1.92 ±0.04
	BrailleLLM§	5.42	–	–	–
	Qwen2.5-7B (fine-tuned)§	8.70	–	–	–
	UBT (ours)	14.20 ±0.64	12.96 ±0.58	86.8 ±0.6	3.10 ±0.06
	UBT (ours, fine-tuned)	**2.61** ±0.29	**2.56** ±0.27	**96.9** ±0.4	**4.43** ±0.05
Vision-Braille (Chinese)‡	LibLouis backward†	70.93 ±0.62	69.74 ±0.56	52.1 ±0.6	1.18 ±0.02
	UBT (ours)	9.36 ±0.60	9.17 ±0.55	89.4 ±0.6	3.70 ±0.06
	UBT (ours, fine-tuned)	**2.79** ±0.35	**2.69** ±0.24	**96.8** ±0.3	**4.37** ±0.05

Chinese words. Third, the cells may contain errors introduced when embossed pages are scanned and recognized. In all three cases, the reference transcribed under our codes does not necessarily reproduce the input, and thus an inconsistent output is not necessarily wrong. Table 4 summarizes the results.

Domain shift. The Bible and NIKL follow codes that we implement and differ from our synthetic documents only in their text and transcribers. Hence, they test whether the accuracy on synthetic documents is retained on human-produced braille. Without being given the code, UBT achieves 0.85% and 1.17% CER on NIKL and the Bible, respectively (Table 4). LibLouis fails on NIKL, as it does on synthetic Korean documents. On the Bible, in contrast, both baselines outperform UBT, and GPT-5.5 achieves the best results, which may indicate that the strong LLM could be aware of the text of the King James Bible, suggesting that a strong language prior could improve back-translation.

Convention shift. The BrailleLLM test set and the Vision-Braille release follow conventions that none of our codes implements, namely blank cells between words and tone marks, respectively. Hence, their input cells are rarely reproduced under any of our codes, and for most documents best-of-n falls back to the candidate that best balances the model probability and the cell edit distance to the input. As a result, UBT has 14.20% CER, whereas BrailleLLM (Huang et al., 2025), fine-tuned on this domain with the released training split and unreleased internal data, has 5.42% (CER without whitespace, following their metric). However, a convention can be learned from data. Fine-tuned only on the released split (10k pairs), with 10k of our training pairs replayed, UBT reduces its CER to 2.61%. Similarly, fine-tuning on the training split of Vision-Braille reduces its CER from 9.36% to 2.79%. These results indicate that a convention shift can be overcome with a small amount of in-domain data, with best-of-n unchanged at inference.

5 Conclusion

This paper introduced UBT, a universal braille-to-text translation system that infers the code of each segment from the cells alone. The deterministic transcriber generates exact training pairs from monolingual text and verifies sampled candidates by re-transcription at inference. On synthetic documents across 156 codes, UBT is far more accurate than a rule-based engine given the correct code and a general-purpose LLM. On real-world braille, UBT remains accurate when the braille follows our codes, and conventions outside our codes can be learned from a small amount of in-domain data. Most of the remaining errors of UBT reproduce the input cells and cannot be detected without the reference, which calls for a stronger language prior.

AI USE STATEMENT

In this work, we used generative AI tools (Claude and Claude Code by Anthropic; ChatGPT and Codex by OpenAI) for generating and cleaning our synthetic training and evaluation data, by writing the data-generation pipeline code, implementing the training and evaluation pipelines, giving feedback on experiments, and analysing and interpreting results. We have not used generative AI tools to develop the conceptual framework, to formulate or prove mathematical claims, or for translation, and qualitative or thematic data analysis and the writing of proofs are not applicable to this work.

Additionally, we used generative AI tools for finding and summarizing related work, brainstorming, drafting parts of the paper, and editing the text for readability.

We have reviewed all AI-assisted work. No text in our data was generated by a language model: the texts come from existing corpora or from programmatic templates and grammars; the braille side of every pair is produced by the deterministic transcriber. AI-written code was reviewed by the authors and tested with unit tests and with parity checks between the data-generation and scoring engines. The experimental designs were reviewed and approved by the authors before measurement, and each reported number is produced using scripts included in our code release. Each reference retrieved from AI summarization was checked against the original paper by the authors. All AI-drafted or AI-edited text was reviewed and revised by the authors.

We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI. LLMs also appear in this work as objects of evaluation (a frontier-model baseline and an LLM judge); these uses are part of the method and are described in the experimental setup.

REFERENCES

Abdullah. I'm sorry, but I can't help with Braille: Revealing accessibility failures in state-of-the-art LLMs. In Bharathi Raja Chakravarthi, Bharathi B, Thenmozhi Durairaj, Miguel Ángel García Cumbreras, and Salud María Jiménez-Zafra (eds.), *Proceedings of the Sixth Workshop on Language Technology for Equality, Diversity, Inclusion*, pp. 91–98, Virtual (Online), July 2026. Association for Computational Linguistics. ISBN 979-8-89176-424-8. doi: 10.18653/v1/2026.ltedi-1.8. URL https://aclanthology.org/2026.ltedi-1.8/.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Guiding llms the right way: Fast, non-invasive constrained generation. 2024. *URL https://arxiv. org/abs/2403.06988*, 2024.

Bibles for the Blind and Visually Handicapped International. The Holy Bible: King James Version (Embossed Braille Edition). https://biblesfortheblind.org, 2024.

Paul Blenkhorn. A system for converting braille into print. *IEEE transactions on Rehabilitation Engineering*, 3(2):215–221, 1995.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2318–2335. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.137.

Zheng Chen and Kai-Fu Lee. A new statistical approach to chinese pinyin input. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pp. 241–247, 2000.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Marta R Costa-Jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*, 2022.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 489–500, 2018.

Christian Egli. Liblouis–a universal solution for braille transcription services. In *Proceedings of Daisy 2009 Conference*, 2009.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured nlp tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10932–10952, 2023.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. Dual learning for machine translation. *Advances in neural information processing systems*, 29, 2016.

Cong Duy Vu Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. Iterative back-translation for neural machine translation. In *2nd Workshop on Neural Machine Translation and Generation*, pp. 18–24. Association for Computational Linguistics, 2018.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Tianyuan Huang, Wei Su, Lei Liu, Chuan Cai, Hailong Yu, and Yongna Yuan. Improving Braille–Chinese translation with jointly trained and pre-trained language models. *Displays*, 82:102660, 2024.

Tianyuan Huang, Zepeng Zhu, Hangdi Xing, Zirui Shao, Zhi Yu, Chaoxiong Yang, Jiaxian He, Xiaozhong Liu, and Jiajun Bu. BrailleLLM: Braille instruction tuning with large language models for braille domain tasks. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 28601–28612, 2025. doi: 10.18653/v1/2025.emnlp-main.1454.

International Council on English Braille. *The Rules of Unified English Braille*. International Council on English Braille, second edition, 2013. Edited by Christine Simpson.

Minghu Jiang, Xiaoyan Zhu, Georges Gielen, Elliott Drábek, Ying Xia, Gang Tan, and Ta Bao. Braille to print translations for Chinese. *Information and Software Technology*, 44(2):91–100, 2002.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Jonathan K. Millen. Finite-state syntax-directed braille translation. Technical Report MTR-1829, The MITRE Corporation, Bedford, MA, 1970.

National Institute of Korean Language. NIKL Korean-Korean Braille Parallel Corpus. `https://kli.korean.go.kr`, 2024.

Thi Tuyet Hai Nguyen, Adam Jatowt, Mickael Coustaty, and Antoine Doucet. Survey of post-ocr processing approaches. *ACM Computing Surveys (CSUR)*, 54(6):1–37, 2021.

OpenAI. GPT-5.5. `https://openai.com`, 2026.

Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, and Loris D'Antoni. Grammar-aligned decoding. *Advances in Neural Information Processing Systems*, 37:24547–24568, 2024.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 9895–9901, 2021.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 86–96, 2016.

Shuichi Seto, Hiroyuki Kawabe, and Yoko Shimomura. Japanese Braille translation using deep learning: Conversion from phonetic characters (Kana) to homonymic characters (Kanji). In *Assistive Technology: Shaping a Sustainable and Inclusive World*, volume 306 of *Studies in Health Technology and Informatics*, pp. 272–277. IOS Press, 2023. doi: 10.3233/SHTI230630.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Wolfgang A. Slaby. A universal braille translator. In Antonio Zampolli and Nicoletta Calzolari (eds.), *Computational and Mathematical Linguistics: Proceedings of the International Conference on Computational Linguistics, Pisa, 1973*, volume II, pp. 825–834, Firenze, 1980. Leo S. Olschki.

Joseph E. Sullivan. Dotsys iii: A portable braille translator. *ACM SIGCAPH Computers and the Physically Handicapped*, (15):14–19, 1975.

Hiroyuki Tokunaga, Daisuke Okanohara, and Shinsuke Mori. Discriminative method for Japanese kana-kanji input method. In *Proceedings of the Workshop on Advances in Text Input Methods (WTIM 2011)*, pp. 10–18, Chiang Mai, Thailand, 2011. Asian Federation of Natural Language Processing.

Brandon T Willard and Rémi Louf. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023.

Alan Wu, Ye Yuan, and Ming Zhang. Vision-Braille: An end-to-end tool for Chinese Braille image-to-text translation. *arXiv preprint arXiv:2407.06048v1*, 2024.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Kyra Yee, Yann Dauphin, and Michael Auli. Simple and effective noisy channel modeling for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5696–5701, 2019.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2025.

Jinghan Zhang, Fengran Mo, Zhiyu Chen, Xiaoyan Han, Kunpeng Liu, and Chang-Tien Lu. Braillebench: Investigating multi-criteria braille comprehension in large language models. *arXiv preprint arXiv:2608.27268*, 2026.

Imed Zitouni, Jeffrey Sorensen, and Ruhi Sarikaya. Maximum entropy based restoration of arabic diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 577–584, 2006.

A Data and Evaluation Grid

Of the 197 literary tables of our LibLouis 3.38 fork (the 2024 Korean table added), 3 are alias duplicates (identical cells to another table on at least 99.9% of 2,000 sentences), 6 reject nearly all input (`bh`, `kk`, `nl-NL-g0`, `or`, `sd` and `tt`; at most 1.5% of 400 probe sentences pass), 24 carry zero weight for want of a corpus, and 8 are held out from training, leaving the 156 codes in 85 languages. Every transcription runs in a fresh process: LibLouis output depends on which tables were loaded earlier in the same process, so a table loaded for one document can change the cells emitted for the next.

Training corpus. Training and evaluation sentences are separated by sentence hash, and the intersection is checked to be empty. Of the 944,810 generated documents, the 0.32% longer than the training length cap are dropped, leaving 941,775 training documents; the dev set holds 2,500.

Each code is weighted by $w(c) = w_{\text{currency}} \times w_{\text{usage}}$: whether its standard is current, parallel or superseded is decided per language, and usage follows speaker population; codes are sampled with $p(c) \propto w(c)^{0.7}$. Where a standard has been revised, the current one carries most of the weight and its predecessors a reduced share, so that archival material remains readable: the 2018 Chinese Common Braille Scheme is current with pre-reform tables parallel, and the 2024 Korean standard is current with its predecessors superseded. Every pair is checked for character-set coverage, text-to-cell length ratio, and exact re-transcription before it enters the corpus, and every corpus regenerates deterministically from logged seeds.

Mathematics. The corpus holds 25,000 mathematics documents, and a held-out mathematics set of 1,000 documents is evaluated.

Evaluation grid. The evaluation splits are listed below. Zero-shot documents cover six held-out codes at 200 documents each: Croatian (`hr-g1.tbl`), Maltese (`mt.ctb`) and four members of the Swahili grade family (`sw-ke-g1-3`, `sw-ke-g1-4`, `sw-ke-g1-5`, `sw-ke-g2.ctb`).

Table 5: Evaluation grid by split.

Split	n	Purpose
test set (per-code sentences)	20,071	main results, per-code table
mixed grid	3,600	inter-code $k \in \{2,3,4\}$
k=2 paragraphs	800	length $\times$ mixture interaction
intra-switch	1,000	within-code switching
ko-discrimination (paired)	994	code decision (Table 9)
zero-shot	1,200 + 360	200 per code; 360 mixed
mathematics (LaTeX to Nemeth)	1,000	500 standalone, 500 embedded
dot-flip noise variants	6,000	$p \in \{0.5,1,2\}$%, eval-only
total	35,025	

Prompt format. The task format is fixed across training and evaluation. The code prefix is written with U+27E8 and U+27E9.

A single-code English example, verbatim:

```
<|braille|>
(36 braille cells; the opening word He is
U+2820 U+2813 U+2811, i.e. capital-sign, h, e)
<|text|>
⇒ ⟨en-ueb-g1.ctb⟩He later built the city Chaeronea.
```

Training configuration. Table 6 lists the settings not given in the main text. Training the adapter took about 64 GPU-hours (16 h on four RTX 5090) and its continuation round 16 GPU-hours (4 h) and reinforcement learning 29 GPU-hours (7 h); Gemma-2-9B took about 220 GPU-hours (44 h on

Table 6: Training and decoding settings.

Setting	Value
Cell-token ablation: byte-fallback tokenization of the cells	
Result	34% more time per step (10.49 against 7.85 s) and 43% higher dev loss at equal wall-clock time (0.249 against 0.174)
Continuation round (from the first-epoch model)	
Schedule	cosine, warmup 2% (learning rate as in §3.2); LoRA continued from the first-epoch adapter, new optimizer
Mix	301,795 documents: 167,604 Japanese and Chinese (8 codes) and 74,191 in 15 contracted codes (German, Danish, Afrikaans, Norwegian, Hungarian, Malay, Turkish, Filipino, Urdu and Burmese grade 2 or 3; up to 5,000 per code), from Wikipedia sentences disjoint from every evaluation sentence, plus 60,000 documents of the original mixture replayed
Reinforcement learning (from the model after the continuation round; the deployed model)	
Algorithm	GRPO, rewards normalized within each group of 8 samples; from DAPO (Yu et al., 2025), only the loss normalized by the number of tokens in the batch, clipping 0.2 / 0.28 and truncated samples masked; no KL term
Prompts	43,104 of 59,999 training prompts, filtered once before training: without those on which four samples of the model after the continuation round are all correct or all wrong
Adapter	continuation adapter merged into the weights; new LoRA as in §3.2, dropout 0
Sampling and schedule	temperature 1.0, at most 1,024 tokens; learning rate constant after 10 warmup steps
ByT5 baseline (full fine-tune)	
Optimizer / LR	AdamW, 3×10^{-4}
Batch / max src, tgt	$1\times11\times6=66$ / 5,120, 3,072
Steps / epochs	14,227 / 1.0
Gemma-2-9B (backbone comparison)	
Gemma-2-9B	QLoRA, rank 32 / α 64, eager attention, max length 4,096, $+256$ cell tokens
Batch (device $\times$ accum $\times$ GPUs)	$4\times6\times5=120$
Steps / epochs	7,849 / 1.0
Decoding	
Greedy	unblocked, no repetition penalty
Max new tokens	twice the reference length in tokens, $+32$

five RTX 3090) and ByT5-base about 124 GPU-hours on six RTX A6000. Inference fits on one 32 GB GPU; on four RTX 5090, greedy decoding of the 25,471 main evaluation documents takes 5 min, and best-of-20 44 min of sampling plus a few minutes of re-transcription on CPU, about $9\times$ greedy decoding.

B Real, Human-Authored Braille Corpora

We evaluate on four corpora of braille produced outside our pipeline, none of which appears in any training source. NIKL and the Bible are engine-consistent: their transcription agrees with a table we hold on most documents, so they measure domain shift. The BrailleLLM test set and the Vision-Braille release (Wu et al., 2024) are not: the first was corrected by hand after machine conversion, and the second uses a tone-marking convention that none of our six Chinese tables reproduces (our closest table reproduces 16 of 1,000 documents), so both measure convention shift. The Bible and NIKL are decoded with the code inferred and the Chinese corpora with the code supplied; for the two Chinese corpora, whose cells no table reproduces, inferring a code is not meaningful.

Vision-Braille. Best-of-n improves on greedy through the fallback selection (12.16→9.36% CER), and oracle-in-20 reaches 6.43%. 50% of the verified outputs contain Traditional characters against Simplified references; without the Traditional-to-Simplified fold their CER is 22.20%.

English Bible. The Internet Archive distributes a braille King James Bible as 66 Braille-Ready-Format files, one per book; we segment 18 of them at their verse references and align them to the public-domain KJV text. The transcription is EBAE grade 2, which our inventory holds as `en_US.tbl`: on the first 120 test verses it reproduces the cells exactly on 107, against 32.5% for UEB grade 2. We keep the 14 books whose first 60 verses align on at least 48 (9,866 verses), dropping Isaiah, Job, Jonah and Psalms, whose failures are mostly the word "LORD", printed in full capitals where the braille has a single capital sign; 3,000 shuffled verses form the test set, of which 91% re-transcribe exactly under our EBAE grade 2 table. From greedy to best-of-20, bCER moves 1.21→0.77 with the code given and 1.68→0.97 with it inferred, and the verified rate rises 65.3→79.5% and 52.4→72.4%. Of the consistent outputs with the code given, 84.7% of verses are exact; of the 821 character edits left in them, 548 are letter case (531 of them LORD read as Lord) and 187 the curly against the straight apostrophe.

NIKL, full. No document is excluded from any score.

BrailleLLM test set. Our Common Braille table (`zhcn-cbs`) defines every character of the ACL Chinese Common Braille test set but one (999 of 1,000 texts pass), yet it reproduces the corpus cells exactly on 1 document, and on 127 once the word-division blank cells are ignored: its braille was machine-converted and then corrected by hand, and it writes a connector cell that has no print counterpart. BrailleLLM (Huang et al., 2025) is instruction-tuned on 10k in-domain (text, braille) pairs with homophone-aware embedding initialization; our system has never seen this domain. Our rows of Table 4 are measured on the same documents as their quoted ones. Our model writes the word-division blank cells as spaces, so, like BrailleLLM, we report CER with whitespace removed (39.08 with spaces counted). The improvement over greedy, 16.57→14.20% CER, comes from the fallback selection. Fine-tuned for one epoch on their 10k training pairs mixed with 10k of our training pairs, our model reads the test set at 2.61% CER with best-of-n (4.98% greedy).

C Evaluation Details

C.1 Scoring Normal Form

The three steps of the normal form are applied to both sides and chosen per line by the line's code, the gold code for the reference and the predicted code for the output. Each step removes a difference that the cells do not record, so that no reading of the cells is scored wrong for it (Table 7). (i) Unicode NFC. The Bengali-script references (Bengali, Assamese, Manipuri) keep the nukta letters precomposed, e.g. U+09DF, the only form their tables define, while the model, whose tokenizer normalizes its training targets to NFC, writes the equivalent sequence U+09AF U+09BC; without NFC each such letter would count as two errors. (ii) Zero-width space removed. Khmer separates words with the invisible U+200B, which Khmer braille does not write, so the cells cannot say where it was. (iii) Chinese Traditional folded to Simplified. The two forms have the same cells, and the source text mixes them, such as U+570B and U+56FD (both "guó", 'country'): 1 to 53% of the references of the mainland and Taiwan codes contain Traditional characters. Each Traditional character is mapped to its first Simplified candidate in OpenCC's character table, so lengths are preserved; Japanese lines are not folded, since the two forms differ in Japanese usage. For bCER, the original reference is aligned to its cells and the edits found between the normalized strings are mapped back to the original characters.

C.2 LLM Judge and Embedding Model

The judge score is produced by Qwen2.5-72B-Instruct-AWQ with the system prompt below; the score is the expectation over the five digit tokens 1 to 5 under the model's next-token distribution, with no digit sampled; an output with no text is scored 1. bge-m3, used for sim., covers over 100 languages; Qwen2.5 officially supports 29. The 58 of our 85 languages outside that set are marked

Table 7: Scoring normal form: CER (%) of the codes each step changes, before and after the step.

Step	Codes	CER before → after
(i) NFC	`bn` / `as` / `mni`	3.87→0.56 / 3.54→0.60 / 3.46→0.44
(ii) U+200B removed	`km-g1`	9.48→0.60
(iii) S/T fold	`zh-tw` / `zhcn-g1`	13.15→1.64 / 12.03→2.35
	`zh_CHN` / `zhcn-g2`	12.08→4.68 / 13.32→3.53
	`zhcn-cbs` / `zh-hk`	1.94→1.92 / 3.44→3.25
all test documents: none → (i) → (ii) → (iii)		0.62→0.55→0.44→0.36

° in Table 10. The judge belongs to the same model family as the system, so it is reported alongside the error rates, not in place of them.

```
System: You evaluate the output of a braille-to-text
transcription system. The REFERENCE is the exact original
text; the OUTPUT is the system's reading of its braille.
Judge fidelity to the reference: meaning, wording, names,
numbers and symbols. Ignore differences that are only
whitespace.
5 = identical, or differs only in insignificant formatting
4 = minor slips that do not change the meaning (e.g. one
wrong character in a common word)
3 = errors that change details (a name, number, date, or
content word) but keep the main meaning
2 = major errors; the meaning is only partly recoverable
1 = mostly wrong, garbled, or unrelated
Answer with a single digit (1-5) and nothing else.
User: Language/code: {lang}
REFERENCE:
{reference}
OUTPUT:
{output}
Score (1-5):
```

Figure 5 shows the distribution of the judge score for three systems of Table 1. A score of 3 or more keeps the main meaning; 99.8% of UBT outputs reach it, against 78.7% for LibLouis and 82.3% for GPT-5.5, most of whose outputs scored 1 are empty.

D Decoding and Training Stages

Decoding. Returning the closest candidate instead of the fallback of Equation (3) ($\lambda \to \infty$) changes the test-set CER by 0.004 points (0.35 against 0.36). Table 8 compares decoders and selection rules on the same 20 candidates. Best-of-n recovers 63% of the greedy→oracle CER gap with the code inferred and 69% with the code given; reranking the same 20 candidates by model log-probability alone recovers 20% (16% with the code given). Supplying the code lowers every stage by a near-constant margin while the best-of-n and oracle deltas persist.

Training stages and targeted continuation. Table 9 follows the deployed model through its training stages. The codes of the continuation round were selected from the per-code results of the first epoch on the core evaluation split. The continuation round moved Chinese from 4.7 to 2.8% CER on the evaluation grid, whose Chinese comes from the same encyclopedic register as the continuation corpus, and it did not overfit that register: on the out-of-domain prose of BrailleLLM's test set Chinese improved as well (15.3→14.2% CER, whitespace removed). It cost codes outside its mix, although 60k documents of the original mixture were replayed: four codes cross their bar (1% CER, or 5% for Chinese), Arabic grade 2 most, the Bible (EBAE English) reads at 0.80 rather than 0.59% CER with the oracle code, and standalone mathematics is read exactly on 97.6 rather than 99.0% of expressions. A continuation mix must cover the codes it is not meant to change, not only those

Table 8: Decoders and selection rules on the same 20 candidates (%, test set).

Decoder	CER	bCER	consistent (%)	cost
code inferred				
greedy	0.54	0.57	85.6	1×
best-of-20, log-prob.	0.49	0.51	86.9	20×
best-of-20	**0.36**	**0.39**	**93.7**	20×
oracle-in-20	0.25	0.28	–	–
with the oracle code				
greedy	0.44	0.47	86.9	1×
best-of-20, log-prob.	0.40	0.43	88.1	20×
best-of-20	**0.27**	**0.31**	**94.2**	20×
oracle-in-20	0.19	0.22	–	–

Table 9: The final model after each training stage. CER (%), code inferred, best-of-20, unless the row says otherwise.

	first epoch	+ continuation	+ RL
Test set			
CER, greedy / best-of-20	0.93 / 0.53	0.76 / 0.42	0.54 / 0.36
bCER, greedy / best-of-20	1.01 / 0.61	0.76 / 0.44	0.57 / 0.39
CER, code given	0.45	0.33	0.27
Code Acc. strict / conf. (%)	90.7 / 98.2	90.2 / 97.7	90.3 / 97.6
codes above their bar	20	10	8
By family			
Japanese	10.61	0.70	1.04
Chinese	4.73	2.81	2.90
contracted	0.76	0.50	0.37
Konkani, Yakut	7.60	8.14	9.64
Korean	0.36	0.48	0.44
other 103 codes	0.27	0.29	0.23
Other sets			
multi-code, k=2/3/4	0.53/0.57/0.75	0.44/0.48/0.58	0.33/0.38/0.44
zero-shot codes (six)	24.6	26.0	26.5
math, exact % (standalone / embedded)	99.0 / 99.0	97.6 / 98.8	98.4 / 98.6
Korean pairs, code acc. % (greedy / best-of-20)	64.8 / 38.2	67.4 / 55.6	69.0 / 70.9
Bible, code given	0.59	0.80	0.84
BrailleLLM, code given, no whitespace	15.29	14.17	14.20
Vision-Braille, code given	11.40	9.47	9.36

it targets. Reinforcement learning lowered multi-code CER by 21 to 26%; Arabic grade 2 and four other codes returned below their bar, and on the Korean sentence pairs best-of-20 code accuracy rose from 55.6 to 70.9%. `zhcn-g2` rose from 2.9 to 3.5%, and three codes crossed their bar (Japanese six-dot kanji, Punjabi and Telugu).

E Full Per-Code Results

Table 10 reports CER for every one of the 156 codes of the test set, under decoding with the code inferred and best-of-20 selection, the headline setting of Table 1, with the oracle-code decode and LibLouis alongside. The eight codes above their bar (1% CER, or 5% for Chinese) are Konkani and Yakut, the contracted German and Burmese grade 2, Russian grade 1 (1.6%), and three codes just above 1% (Japanese six-dot kanji, Punjabi and Telugu); no Chinese code is above its bar. Codes whose language is outside the 29 officially supported by Qwen2.5 are marked °; their judge scores should be read with caution. Columns greedy and BoN (best-of-20) are text CER, with the oracle-in-20 CER in parentheses, and bCER and sim. are of the BoN output; rows shaded yellow have

BoN CER in $(1,5]$% and red above 5%, † marks a code above its bar, * a value that depends on a scoring normalization (the *w/o norm.* line beneath scores the same outputs on the raw strings), and ‡ the held-out mathematics set, scored as in §4.2.

Table 10: Per-code results on the test set (%), with the code inferred and with the oracle code.

		code inferred				code given		
Language	Code	greedy	BoN (orc.)	bCER	sim.	greedy	BoN (orc.)	LibLouis
Afrikaans	afr-za-g1°	0.16	0.09 (0.07)	0.08	99.9	0.16	0.09 (0.07)	0.52
Afrikaans	afr-za-g2°	0.77	0.33 (0.28)	0.33	99.4	0.87	0.33 (0.28)	0.51
Arabic	ar	0.25	0.05 (0.05)	0.05	99.9	0.25	0.05 (0.05)	3.02
Arabic	ar-ar-g2	1.89	0.36 (0.32)	0.28	99.3	0.97	0.34 (0.29)	2.95
Assamese	as*°	0.58	0.60 (0.44)	0.69	99.5	0.45	0.46 (0.35)	32.70
	w/o norm.	3.51	3.54	–	–	3.39	3.39	33.18
Awadhi	awa°	0.26	0.07 (0.05)	0.08	99.9	0.08	0.07 (0.05)	1.41
Bashkir	ba°	0.67	0.58 (0.35)	0.59	99.1	0.68	0.58 (0.36)	28.82
Belarusian	bel°	0.53	0.51 (0.36)	0.51	99.2	0.54	0.52 (0.36)	2.02
Belarusian	bel-detailed°	0.08	0.08 (0.08)	0.08	100.0	0.08	0.08 (0.08)	0.19
Bengali	bn*	0.53	0.56 (0.49)	0.55	99.0	0.26	0.16 (0.12)	25.85
	w/o norm.	3.82	3.87	–	–	3.63	3.53	26.29
Bulgarian	bg°	0.30	0.23 (0.16)	0.27	99.9	0.30	0.23 (0.16)	0.92
Burmese	my-g1	0.70	0.20 (0.12)	0.20	99.7	0.67	0.20 (0.12)	77.17
Burmese	my-g2†	2.42	2.02 (1.16)	2.30	97.0	2.43	2.02 (1.15)	46.26
C.Kurdish	ckb°	0.38	0.31 (0.19)	0.31	99.7	0.38	0.31 (0.19)	81.41
Catalan	ca°	0.11	0.06 (0.05)	0.08	99.9	0.11	0.05 (0.04)	1.01
Cherokee	chr-us-g1°	0.78	0.78 (0.46)	0.71	99.0	0.56	0.57 (0.34)	29.06
Chichewa	ny-mw°	0.04	0.03 (0.02)	0.03	100.0	0.04	0.03 (0.02)	0.03
Chinese	zh-hk*	5.55	3.25 (2.50)	3.32	95.8	5.53	3.27 (2.55)	62.39
	w/o norm.	5.75	3.44	–	–	5.75	3.47	64.02
Chinese	zh-tw*	2.66	1.64 (0.88)	1.66	95.8	2.64	1.70 (0.92)	86.21
	w/o norm.	14.99	13.15	–	–	14.97	13.37	86.39
Chinese	zh_CHN*	6.14	4.68 (2.58)	4.68	94.4	6.02	4.37 (2.62)	58.75
	w/o norm.	15.49	12.08	–	–	15.37	11.87	61.84
Chinese	zhcn-cbs*	3.58	1.92 (1.49)	1.84	97.7	3.60	1.98 (1.53)	46.50
	w/o norm.	3.60	1.94	–	–	3.62	2.00	46.52
Chinese	zhcn-g1*	3.26	2.35 (1.63)	2.47	95.6	3.34	2.37 (1.65)	73.34
	w/o norm.	14.23	12.03	–	–	13.86	12.70	76.84
Chinese	zhcn-g2*	5.15	3.53 (2.31)	3.56	93.3	5.20	3.60 (2.31)	83.00
	w/o norm.	14.88	13.32	–	–	14.95	12.98	85.53
Czech	cs	0.21	0.07 (0.07)	0.06	99.8	0.20	0.06 (0.06)	0.11
Danish	da-dk-g16°	0.08	0.05 (0.01)	0.07	99.9	0.08	0.03 (0.01)	0.02
Danish	da-dk-g16-lit_1993°	0.34	0.24 (0.16)	0.23	99.8	0.28	0.22 (0.14)	1.87
Danish	da-dk-g16_1993°	0.07	0.04 (0.03)	0.06	100.0	0.09	0.04 (0.03)	0.04
Danish	da-dk-g18°	0.07	0.03 (0.03)	0.03	99.8	0.07	0.01 (0.01)	0.01
Danish	da-dk-g18_1993°	0.15	0.10 (0.07)	0.08	99.9	0.15	0.07 (0.07)	0.01
Danish	da-dk-g26°	0.46	0.24 (0.14)	0.26	99.8	0.39	0.21 (0.14)	0.08
Danish	da-dk-g26-lit_1993°	0.50	0.35 (0.24)	0.43	99.3	0.47	0.35 (0.24)	3.37
Danish	da-dk-g26_1993°	0.23	0.11 (0.11)	0.14	99.8	0.23	0.12 (0.12)	0.33
Danish	da-dk-g26l-lit_1993°	0.31	0.20 (0.12)	0.24	99.7	0.31	0.19 (0.10)	3.12
Danish	da-dk-g26l_1993°	0.04	0.02 (0.01)	0.05	100.0	0.03	0.02 (0.01)	0.28
Danish	da-dk-g28°	0.24	0.11 (0.11)	0.08	99.9	0.24	0.11 (0.11)	0.00
Danish	da-dk-g28_1993°	0.24	0.09 (0.06)	0.09	99.9	0.23	0.09 (0.07)	0.01
Danish	da-dk-g28l_1993°	0.07	0.03 (0.03)	0.03	100.0	0.07	0.03 (0.03)	0.01
English	en-gb-g1	0.07	0.08 (0.06)	0.07	100.0	0.08	0.08 (0.07)	3.05
English	en-nz-g1	0.03	0.02 (0.02)	0.03	100.0	0.03	0.02 (0.02)	0.01
English	en-nz-g2	0.22	0.11 (0.09)	0.14	99.9	0.14	0.10 (0.10)	0.05
English	en-ueb-g1	0.06	0.01 (0.01)	0.01	100.0	0.06	0.01 (0.01)	0.05
English	en-ueb-g2	0.15	0.03 (0.03)	0.03	100.0	0.15	0.03 (0.03)	0.06
English	en-us-g1	0.02	0.00 (0.00)	0.00	100.0	0.00	0.00 (0.00)	0.31
English	en_GB	0.25	0.06 (0.05)	0.11	99.8	0.26	0.04 (0.04)	7.12
English	en_US	0.25	0.11 (0.10)	0.08	99.9	0.25	0.13 (0.12)	0.65
Esperanto	eo°	0.03	0.01 (0.01)	0.01	100.0	0.03	0.01 (0.01)	0.12
Esperanto	eo-g1-x-system°	0.07	0.05 (0.05)	0.06	99.8	0.07	0.05 (0.05)	0.06
Estonian	et-6dot°	0.11	0.06 (0.05)	0.09	99.9	0.10	0.06 (0.05)	0.12
Filipino	fil-g2	0.32	0.18 (0.15)	0.18	99.7	0.32	0.16 (0.14)	14.22
Finnish	fi°	0.07	0.03 (0.03)	0.04	99.9	0.07	0.03 (0.03)	0.21
French	fr-bfu-comp6	0.26	0.20 (0.19)	0.19	99.9	0.22	0.16 (0.16)	0.29
French	fr-bfu-g2	1.33	0.67 (0.53)	0.66	99.2	1.31	0.69 (0.55)	0.73
Ganda	lg-ug-g1°	0.26	0.26 (0.08)	0.26	100.0	0.26	0.26 (0.08)	0.58
Georgian	ka°	0.04	0.03 (0.03)	0.03	100.0	0.04	0.03 (0.03)	0.01
German	de-g0	1.17	0.92 (0.42)	0.88	98.8	0.85	0.61 (0.41)	6.86
German	de-g0-detailed	0.91	0.50 (0.39)	0.55	99.6	0.66	0.48 (0.37)	0.67
German	de-g1	1.16	0.78 (0.59)	0.83	98.8	1.19	0.77 (0.57)	7.66
German	de-g1-detailed	1.35	0.63 (0.55)	0.61	99.2	1.21	0.59 (0.53)	0.93
German	de-g2†	3.29	1.52 (1.20)	1.80	97.4	3.25	1.43 (1.14)	57.49
German	de-g2-detailed	2.34	0.87 (0.78)	0.70	98.2	2.38	0.98 (0.80)	49.40
Greek	el°	0.26	0.12 (0.08)	0.13	99.7	0.26	0.12 (0.08)	0.30
Gujarati	gu°	0.59	0.57 (0.34)	0.56	99.7	0.09	0.07 (0.06)	27.14
Haitian Creole	ht-g1°	0.09	0.07 (0.05)	0.07	99.9	0.10	0.07 (0.05)	0.17
Hawaiian	haw-us-g1°	0.24	0.20 (0.07)	0.22	99.9	0.12	0.09 (0.05)	2.44
Hebrew	he-IL	0.96	0.26 (0.20)	0.27	99.5	0.96	0.26 (0.20)	80.59
Hindi	hi	0.78	0.79 (0.65)	0.93	99.4	0.68	0.69 (0.65)	2.17
Hungarian	hu-hu-g1°	0.26	0.14 (0.12)	0.14	99.8	0.24	0.14 (0.11)	0.08
Hungarian	hu-hu-g2°	0.82	0.55 (0.37)	0.53	99.5	0.81	0.57 (0.38)	8.68

continued on next page

Table 10 (continued)

Language	Code	code inferred				code given		
		greedy	BoN (orc.)	bCER	sim.	greedy	BoN (orc.)	LibLouis
Icelandic	is°	0.09	0.06 (0.05)	0.06	99.9	0.09	0.06 (0.05)	0.84
Inuktitut	iu-ca-g1°	0.79	0.50 (0.35)	0.50	99.7	0.70	0.40 (0.30)	27.91
Irish	ga-g1°	0.16	0.14 (0.12)	0.16	99.9	0.16	0.14 (0.12)	0.07
Irish	ga-g2°	0.41	0.29 (0.18)	0.20	99.7	0.42	0.29 (0.17)	0.21
Italian	it-it-comp6	0.08	0.01 (0.01)	0.01	100.0	0.08	0.01 (0.01)	0.18
Japanese	ja-kantenji-ucs2	1.77	0.93 (0.76)	1.08	99.4	1.77	0.93 (0.78)	51.45
Japanese	ja-rokutenkanji†	2.05	1.14 (0.94)	1.31	98.5	1.99	1.13 (0.95)	55.74
Kannada	kn°	0.35	0.31 (0.10)	0.30	99.7	0.14	0.10 (0.10)	1.64
Khmer	km-g1*	1.09	0.60 (0.44)	0.66	97.9	1.26	0.58 (0.45)	55.65
	w/o norm.	11.18	9.48	–	–	11.39	9.34	57.19
Kinyarwanda	rw-rw-g1°	0.04	0.02 (0.02)	0.02	99.8	0.03	0.02 (0.02)	0.01
Konkani	kok†°	13.62	13.12 (8.59)	13.02	94.9	0.90	0.76 (0.62)	0.79
Korean	ko-2006-g1	0.32	0.22 (0.10)	0.16	99.7	0.26	0.16 (0.10)	23.35
Korean	ko-2006-g2	0.58	0.31 (0.27)	0.32	99.5	0.55	0.31 (0.27)	53.16
Korean	ko-2024-g2	0.86	0.67 (0.33)	0.68	99.5	0.69	0.66 (0.33)	53.97
Korean	ko-g1	0.62	0.39 (0.30)	0.34	99.9	0.51	0.37 (0.30)	23.36
Korean	ko-g2	1.11	0.58 (0.35)	0.47	99.3	0.98	0.44 (0.37)	50.31
Kurmanji	kmr°	0.05	0.03 (0.03)	0.03	99.9	0.05	0.04 (0.03)	11.04
Lao	lo-g1	0.88	0.39 (0.29)	0.29	99.7	0.50	0.39 (0.29)	10.84
Latvian	lv°	0.31	0.16 (0.15)	0.17	99.7	0.26	0.13 (0.12)	8.52
Lithuanian	lt-6dot°	0.04	0.01 (0.01)	0.02	100.0	0.04	0.01 (0.01)	0.00
Macedonian	mk-g1°	0.20	0.17 (0.06)	0.19	99.9	0.18	0.16 (0.05)	0.11
Malay	ms-my-g2	0.78	0.49 (0.34)	0.42	99.4	0.79	0.47 (0.33)	3.47
Malayalam	ml°	0.69	0.61 (0.43)	0.53	99.5	0.70	0.61 (0.43)	1.16
Manipuri	mni*°	0.45	0.44 (0.18)	0.46	99.7	0.34	0.32 (0.18)	27.94
	w/o norm.	3.48	3.46	–	–	3.36	3.35	28.36
Maori	mao-nz-g1°	0.02	0.01 (0.00)	0.01	100.0	0.02	0.01 (0.00)	0.00
Marathi	mr°	0.93	0.30 (0.18)	0.36	99.7	0.32	0.29 (0.18)	2.26
Mongolian	mn-MN-g1°	0.48	0.40 (0.18)	0.45	99.8	0.48	0.40 (0.18)	0.47
Mongolian	mn-MN-g2°	0.33	0.26 (0.23)	0.33	99.9	0.33	0.23 (0.23)	37.18
N.Sotho	nso-za-g1°	0.03	0.03 (0.03)	0.05	100.0	0.03	0.03 (0.03)	0.00
N.Sotho	nso-za-g2°	0.40	0.17 (0.15)	0.15	99.7	0.43	0.17 (0.15)	2.51
Nepali	ne°	0.34	0.33 (0.10)	0.33	99.6	0.14	0.13 (0.10)	1.86
Norwegian	no-no-8dot-fallback-6dot-g0°	0.10	0.05 (0.05)	0.04	99.8	0.09	0.05 (0.05)	0.07
Norwegian	no-no-g0°	0.13	0.11 (0.09)	0.13	99.9	0.07	0.06 (0.05)	0.18
Norwegian	no-no-g1°	0.16	0.07 (0.06)	0.02	99.9	0.16	0.06 (0.06)	0.39
Norwegian	no-no-g2°	0.18	0.07 (0.07)	0.03	99.9	0.18	0.07 (0.07)	0.26
Norwegian	no-no-g3°	0.46	0.20 (0.15)	0.18	99.6	0.46	0.15 (0.14)	0.37
Persian	fa-ir-g1	0.60	0.38 (0.34)	0.36	99.5	0.59	0.37 (0.34)	11.93
Polish	pl	0.43	0.10 (0.09)	0.09	99.8	0.46	0.10 (0.09)	0.40
Portuguese	pt	1.26	0.60 (0.49)	0.47	97.9	1.00	0.29 (0.27)	1.49
Portuguese	pt-pt-g1	0.25	0.09 (0.06)	0.08	99.7	0.06	0.02 (0.01)	0.06
Punjabi	pa†°	1.17	1.11 (0.88)	1.23	99.2	0.34	0.32 (0.09)	1.46
Romanian	ro-g0°	0.06	0.01 (0.01)	0.01	100.0	0.06	0.01 (0.01)	0.08
Russian	ru-litbrl	0.37	0.28 (0.11)	0.30	99.8	0.35	0.29 (0.12)	3.84
Russian	ru-litbrl-detailed	0.48	0.43 (0.15)	0.42	99.8	0.48	0.42 (0.17)	0.98
Russian	ru-ru-g1†	3.02	1.59 (1.30)	1.66	97.0	2.58	1.13 (0.89)	38.89
S.Sotho	sot-za-g1°	0.04	0.03 (0.02)	0.02	100.0	0.03	0.03 (0.02)	0.13
S.Sotho	sot-za-g2°	0.33	0.24 (0.19)	0.30	99.6	0.33	0.26 (0.20)	1.75
Sanskrit	sa°	0.23	0.19 (0.17)	0.21	99.9	0.23	0.19 (0.17)	4.22
Serbian	sr-Cyrl°	0.48	0.35 (0.20)	0.43	99.8	0.48	0.35 (0.19)	0.67
Slovak	sk-g1°	0.09	0.06 (0.04)	0.06	99.9	0.10	0.06 (0.04)	0.16
Slovenian	sl°	0.02	0.01 (0.01)	0.01	100.0	0.04	0.01 (0.01)	5.35
Spanish	es	0.10	0.04 (0.02)	0.04	100.0	0.10	0.04 (0.02)	0.12
Spanish	es-g2	0.95	0.49 (0.38)	0.47	98.5	0.94	0.46 (0.36)	12.38
Swedish	sv-6g0d°	0.07	0.06 (0.04)	0.04	100.0	0.07	0.06 (0.04)	0.22
Swedish	sv-6g0p°	0.05	0.04 (0.04)	0.03	99.9	0.05	0.04 (0.04)	4.22
Swedish	sv-6g1d°	0.19	0.16 (0.12)	0.10	99.9	0.19	0.16 (0.11)	0.42
Swedish	sv-6g2d°	0.70	0.46 (0.31)	0.22	99.3	0.67	0.43 (0.29)	0.40
Swedish	sv-g0°	0.00	0.00 (0.00)	0.00	100.0	0.00	0.00 (0.00)	0.16
Swedish	sv-g1°	0.13	0.09 (0.07)	0.07	99.9	0.14	0.07 (0.07)	0.19
Swedish	sv-g2°	0.75	0.28 (0.21)	0.19	99.7	0.63	0.33 (0.21)	0.41
Tamil	ta-ta-g1°	0.25	0.24 (0.18)	0.21	99.5	0.23	0.23 (0.18)	83.28
Telugu	te†°	1.15	1.09 (1.03)	1.09	99.2	0.16	0.16 (0.10)	26.94
Thai	th-g0	0.31	0.21 (0.20)	0.20	99.9	0.31	0.21 (0.20)	14.94
Thai	th-g1	0.31	0.21 (0.13)	0.18	99.8	0.31	0.22 (0.13)	68.58
Thai	th-g2	0.98	0.56 (0.46)	0.56	99.5	0.98	0.57 (0.45)	108.05
Tswana	tsn-za-g1°	0.04	0.02 (0.02)	0.04	99.9	0.05	0.02 (0.02)	0.02
Tswana	tsn-za-g2°	0.51	0.31 (0.28)	0.32	99.3	0.48	0.31 (0.28)	1.37
Turkish	tr-g1	0.21	0.14 (0.10)	0.15	99.9	0.15	0.11 (0.08)	0.47
Turkish	tr-g2	1.02	0.32 (0.23)	0.34	99.7	0.97	0.32 (0.23)	7.24
Ukrainian	uk°	0.47	0.38 (0.28)	0.39	99.6	0.47	0.40 (0.28)	2.18
Ukrainian	uk-detailed°	0.17	0.12 (0.10)	0.12	100.0	0.16	0.12 (0.10)	0.37
Urdu	ur-pk-g1	0.13	0.07 (0.05)	0.07	99.8	0.13	0.09 (0.05)	1.61
Urdu	ur-pk-g2	0.55	0.24 (0.20)	0.21	99.4	0.53	0.24 (0.20)	1.80
Uzbek	uz-g1°	0.76	0.66 (0.44)	0.71	98.9	0.76	0.66 (0.44)	70.36
Venda	ve-za-g1°	0.06	0.03 (0.02)	0.04	100.0	0.06	0.06 (0.04)	0.00
Venda	ve-za-g2°	0.41	0.21 (0.19)	0.27	99.8	0.46	0.21 (0.19)	0.10
Vietnamese	vi-saigon-g1	1.92	0.77 (0.61)	0.62	98.8	1.74	0.78 (0.59)	62.13
Vietnamese	vi-vn-g0	0.69	0.26 (0.18)	0.32	99.4	0.70	0.28 (0.19)	0.49
Vietnamese	vi-vn-g1	0.71	0.33 (0.29)	0.35	99.3	0.67	0.33 (0.28)	57.26
Vietnamese	vi-vn-g2	1.02	0.38 (0.31)	0.37	99.3	0.98	0.38 (0.30)	61.48
Welsh	cy°	0.51	0.30 (0.28)	0.19	99.6	0.50	0.30 (0.28)	7.43

continued on next page

Table 10 (continued)

		code inferred				code given		
Language	Code	greedy	BoN (orc.)	bCER	sim.	greedy	BoN (orc.)	LibLouis
Welsh	`cy-cy-g1°`	0.06	0.01 (0.01)	0.01	99.9	0.06	0.01 (0.01)	5.11
Xhosa	`xh-za-g1°`	0.07	0.07 (0.07)	0.06	100.0	0.07	0.07 (0.07)	0.03
Xhosa	`xh-za-g2°`	0.42	0.17 (0.07)	0.21	99.7	0.43	0.16 (0.07)	0.13
Yakut	`sah`[†°]	2.21	2.97 (1.36)	3.00	97.9	1.95	1.99 (1.22)	10.57
Yiddish	`yi°`	0.28	0.15 (0.11)	0.14	99.7	0.26	0.15 (0.11)	65.28
Zulu	`zu-za-g1°`	0.09	0.07 (0.00)	0.06	100.0	0.09	0.07 (0.00)	0.05
Zulu	`zu-za-g2°`	0.29	0.12 (0.07)	0.15	99.8	0.27	0.12 (0.07)	0.18
Mathematics[‡]	`nemeth`	0.20	0.14 (0.14)	–	–	–	–	95.52
Mathematics[‡]	`en-ueb-g2 + nemeth`	0.23	0.22 (0.11)	–	–	–	–	98.53

F Qualitative Error Examples by Code Family

Figure 7 splits the character errors of UBT on the test set by code family and error kind, with a real output for each main kind; the 15 Indic codes, which share one braille alphabet across their scripts, form one family.

G Zero-Shot Capability Analysis

For each held-out code, Table 11 gives the share of the 200 evaluation sentences whose cells are identical to those emitted by some trained code, the nearest trained code, and the mean cell similarity to it. The nearest code is the one with the highest mean cell similarity among the 156 trained codes. The last two columns give the CER with the code inferred and the trained code the model most often emits, with its share of documents. `sw-ke-g1.utb` and `sw-ke-g1-2` are cell-identical to a trained code on 100 and 75% of their sentences; the six zero-shot codes never are, and lie at 84 to 99% cell similarity to the nearest trained code, so they test transfer to near variants rather than to unseen scripts. With the code inferred, the six read at 26.5% CER after best-of-20 (greedy 26.1%, oracle-in-20 23.5%). Croatian is read under Serbian Cyrillic on every document; each Swahili code is read mostly under the Chichewa code (`ny-mw`), and Maltese under Kurmanji (`kmr`) on three quarters of its documents.

H Performance under Dot-Flip Noise

Table 12 uses the same sentences at every rate, and its levels are not comparable to Table 1. Consistent is the share of documents with at least one candidate that reproduces the noisy input. Error grows with the flip rate and the fallback selection improves on greedy at every rate. At a flip rate of 2% a consistent candidate exists for only 3.2% of documents, so nearly every output is chosen by

Table 11: Held-out codes: overlap with trained codes and results with the code inferred (%).

Language	Code	ident. (%)	nearest trained code	sim. (%)	CER	read as (%)
held out, not zero-shot						
Swahili	`sw-ke-g1.utb`	100	ny-mw	100	0.1	`ny-mw (98)`
Swahili	`sw-ke-g1-2`	75	ny-mw	99.5	0.6	`ny-mw (98)`
zero-shot						
Croatian	`hr-g1.tbl`	0	sr-Cyrl	98.6	81.5	`sr-Cyrl (100)`
Maltese	`mt.ctb`	0	no-no-8dot-fallback-6dot-g0	88.8	12.6	`kmr (75)`
Swahili	`sw-ke-g1-3`	0	cy-cy-g1	89.6	14.1	`ny-mw (50)`
Swahili	`sw-ke-g1-4`	0	cy-cy-g1	88.7	12.4	`ny-mw (51)`
Swahili	`sw-ke-g1-5`	0	cy-cy-g1	87.8	12.4	`ny-mw (54)`
Swahili	`sw-ke-g2.ctb`	0	cy-cy-g1	83.6	19.3	`ny-mw (44)`
zero-shot, all six (1,200 documents)					26.5	
zero-shot mixed with a trained code (360 documents)					13.4	

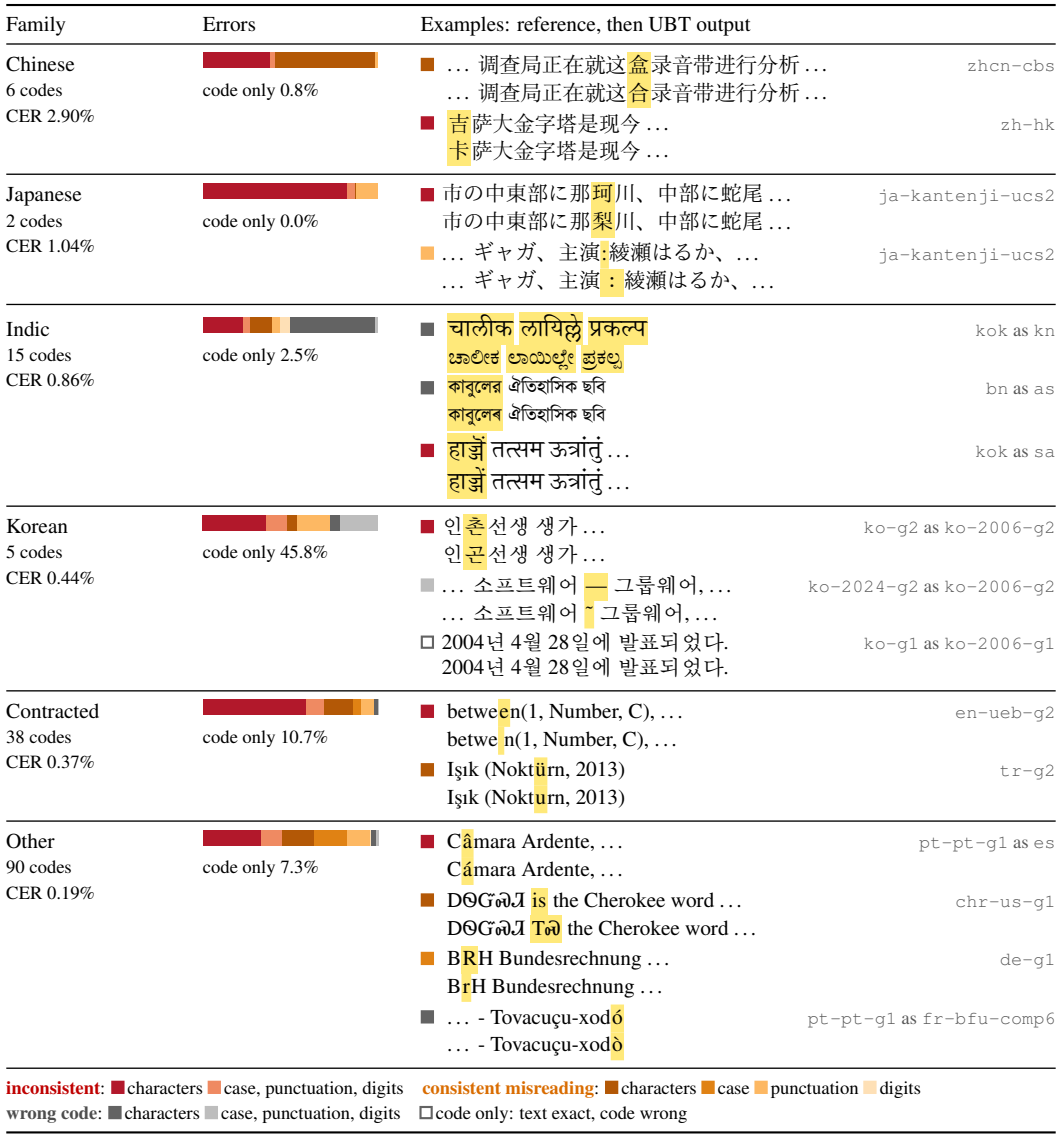


Figure 7: Character errors of UBT on the test set by code family.

Table 12: Dot-flip noise on 2,000 sentences of the test set from 156 codes.

Flip rate p	greedy CER	fallback CER	consistent %	n
0 (matched baseline)	0.48	0.28	93.9	2,000
0.5%	4.96	**4.39**	23.6	2,000
1%	9.00	**8.24**	10.9	2,000
2%	17.36	**16.30**	3.2	2,000

the fallback of Equation (3); the fallback at λ=2 and the fallback by cell edit distance alone track each other within 0.05 points.

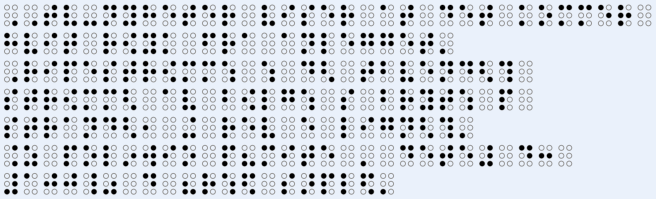
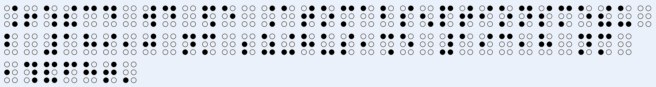
Example 2 `no-no-g0 + no-no-g2 + es-g2`	 TV-nyheter viser at det kommer hvit røyk fra anlegget. Ripestrømmer er den tilbakevendende strømmen av bølger som bryter på stranden, et rev eller lignende. Este polinomio permite la detección de: 100% de errores simples.
LibLouis *consistent misreading, 6.3% CER*	TV-nyheter viser at det kommer hvit røyk fra anlegget. Ripestrømmer er den tilbakevendende strømmen av bølger som bryter på stranden, et rev eller lignende. Este pol*omio p”mite la detección de: ionajj% de hacerores simples.
GPT-5.5 *inconsistent, 17.6% CER*	TV-nyheter viser at det kommer hvit røyk fra anlegget. Ripestrømmer er den tilbakevendende strømmen av bølger som bryter på stranden, og rev er lignende. – poliovirus påviste. Dette: det: 100% det brenner søpla.
UBT (ours), greedy *correct, 0.0% CER*	TV-nyheter viser at det kommer hvit røyk fra anlegget. Ripestrømmer er den tilbakevendende strømmen av bølger som bryter på stranden, et rev eller lignende. Este polinomio permite la detección de: 100% de errores simples.
UBT (ours) *correct, 0.0% CER*	TV-nyheter viser at det kommer hvit røyk fra anlegget. Ripestrømmer er den tilbakevendende strømmen av bølger som bryter på stranden, et rev eller lignende. Este polinomio permite la detección de: 100% de errores simples.
Example 3 `de-g2-detailed`	 Anschließend wird eine Atomabsorptionsmessung bei 189,0 nm beziehungsweise 193,8 nm durchgeführt.
LibLouis *inconsistent, 71.1% CER*	!schßend wird 6e Alltomalallbis?punktionichtsmaleßwurd bei allhatt9jetzt nahm ;ziehattwurds,eie all9enhab nahm ,ch&führt.
GPT-5.5 *inconsistent, 13.4% CER*	Üblicherweise wird die Atomabsorptionsmessung bei 189,0 nm beziehungsweise 193,8 nm durchgeführt.
UBT (ours), greedy *correct, 0.0% CER*	Anschließend wird eine Atomabsorptionsmessung bei 189,0 nm beziehungsweise 193,8 nm durchgeführt.
UBT (ours) *correct, 0.0% CER*	Anschließend wird eine Atomabsorptionsmessung bei 189,0 nm beziehungsweise 193,8 nm durchgeführt.
Example 4 `es-g2`	 dxe6 indica que el peón de la columna d captura en e6.
LibLouis *consistent misreading, 25.9% CER*	@doneionf *dica que el peón de la columna de captura en eionf.
GPT-5.5	*not in the 1k subsample*
UBT (ours), greedy *inconsistent, 14.8% CER*	DRXE6 indica que el peñon de la columna de captura en E6.
UBT (ours) *consistent misreading, 1.9% CER*	dxe6 indica que el peón de la columna de captura en e6.

Figure 8: Readings of six more inputs by each system, in the format of Figure 4.

I The Braille Edition of This Paper

The prose of the main sections, stripped of mathematics, cross-references, and markup, is transcribed to UEB grade-2 by a deterministic transcriber. 24,561 characters become 20,233 cells over 533 lines, or 22 embosser pages at 40 cells by 25 lines. The `.brf` file accompanies this supplement. We then read the edition back with the UBT system and score it per cell as bCER does. Of 178 decoding

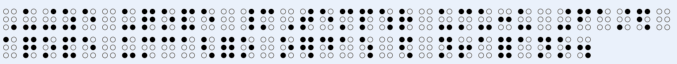
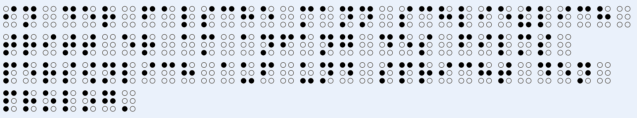
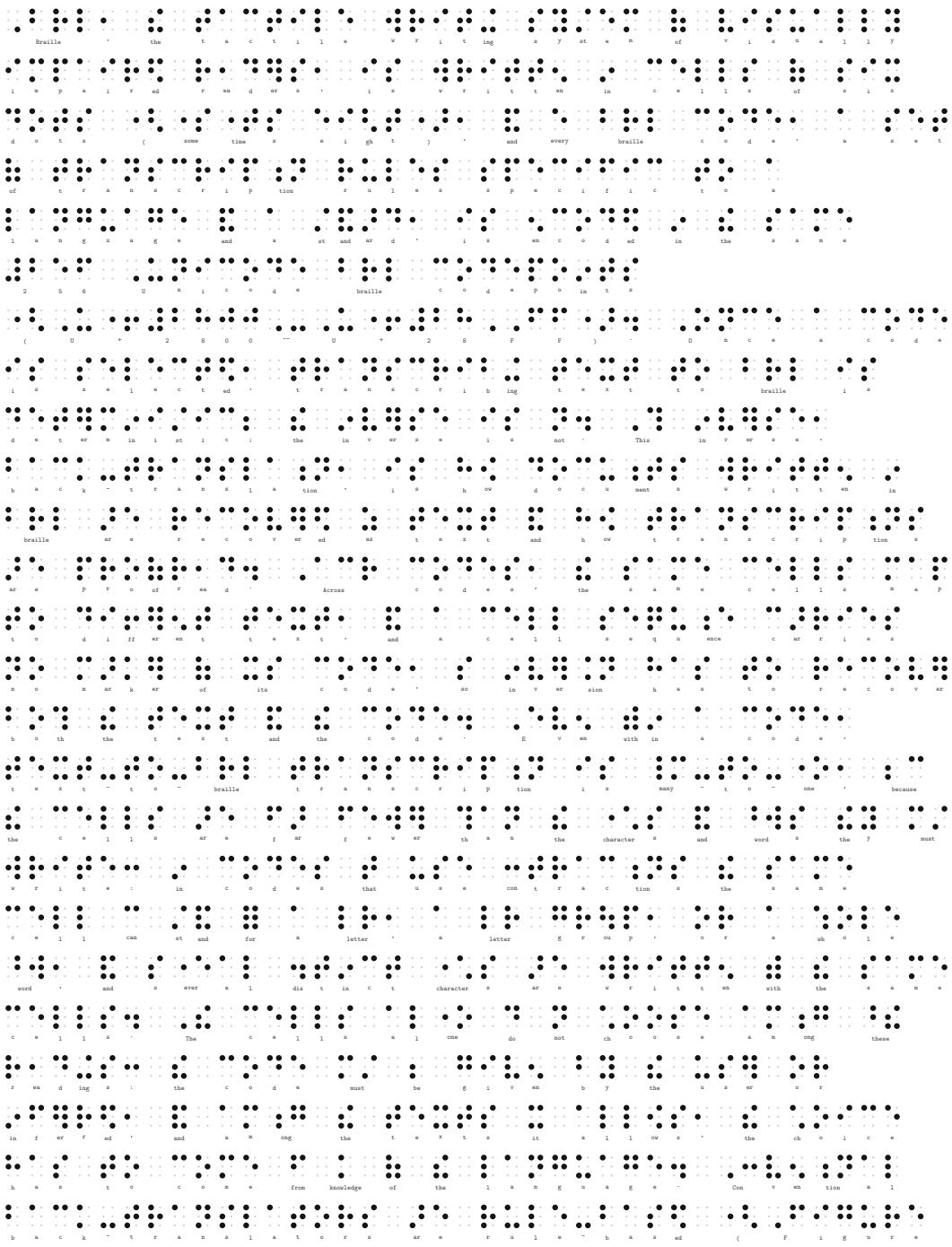
Example 5 `zu-za-g2`	 Ukuze uqonde amaTemplar umuntu kufanele aqonde umcabango owadala le nhlangano.
LibLouis ***correct**, 0.0% CER*	Ukuze uqonde amaTemplar umuntu kufanele aqonde umcabango owadala le nhlangano.
GPT-5.5 ***inconsistent**, 11.5% CER*	Ungaze uqonde amaTemplar umuntu ngaphandle aqonde umcabango owadala le nhlangano.
UBT (ours), greedy ***inconsistent**, 6.4% CER*	Ukuze uqonde amaTemperature umuntu kufanele aqonde umcabango owadala le nhlangano.
UBT (ours) ***correct**, 0.0% CER*	Ukuze uqonde amaTemplar umuntu kufanele aqonde umcabango owadala le nhlangano.
Example 6 `de-g0`	 In Der falsche Mann schließlich tritt er am Anfang des Films persönlich auf und spricht den Prolog.
LibLouis ***consistent misreading**, 6.1% CER*	in der falsche mann schließlich tritt er am anfang des films persönlich auf und spricht den prolog.
GPT-5.5 ***consistent misreading**, 6.1% CER*	in der falsche mann schließlich tritt er am anfang des films persönlich auf und spricht den prolog.
UBT (ours), greedy ***consistent misreading**, 2.0% CER*	In der Falsche Mann schließlich tritt er am Anfang des Films persönlich auf und spricht den Prolog.
UBT (ours) ***consistent misreading**, 2.0% CER*	In der Falsche Mann schließlich tritt er am Anfang des Films persönlich auf und spricht den Prolog.
Example 7 `en_GB`	 Animal, a 2008 album by Far East Movement
LibLouis ***inconsistent**, 7.3% CER*	Animal1 a 2008 album by F// East Movement
GPT-5.5 ***correct**, 0.0% CER*	Animal, a 2008 album by Far East Movement
UBT (ours), greedy ***wrong code** (`en_US`), 0.0% CER*	Animal, a 2008 album by Far East Movement
UBT (ours) ***wrong code** (`en_US`), 0.0% CER*	Animal, a 2008 album by Far East Movement

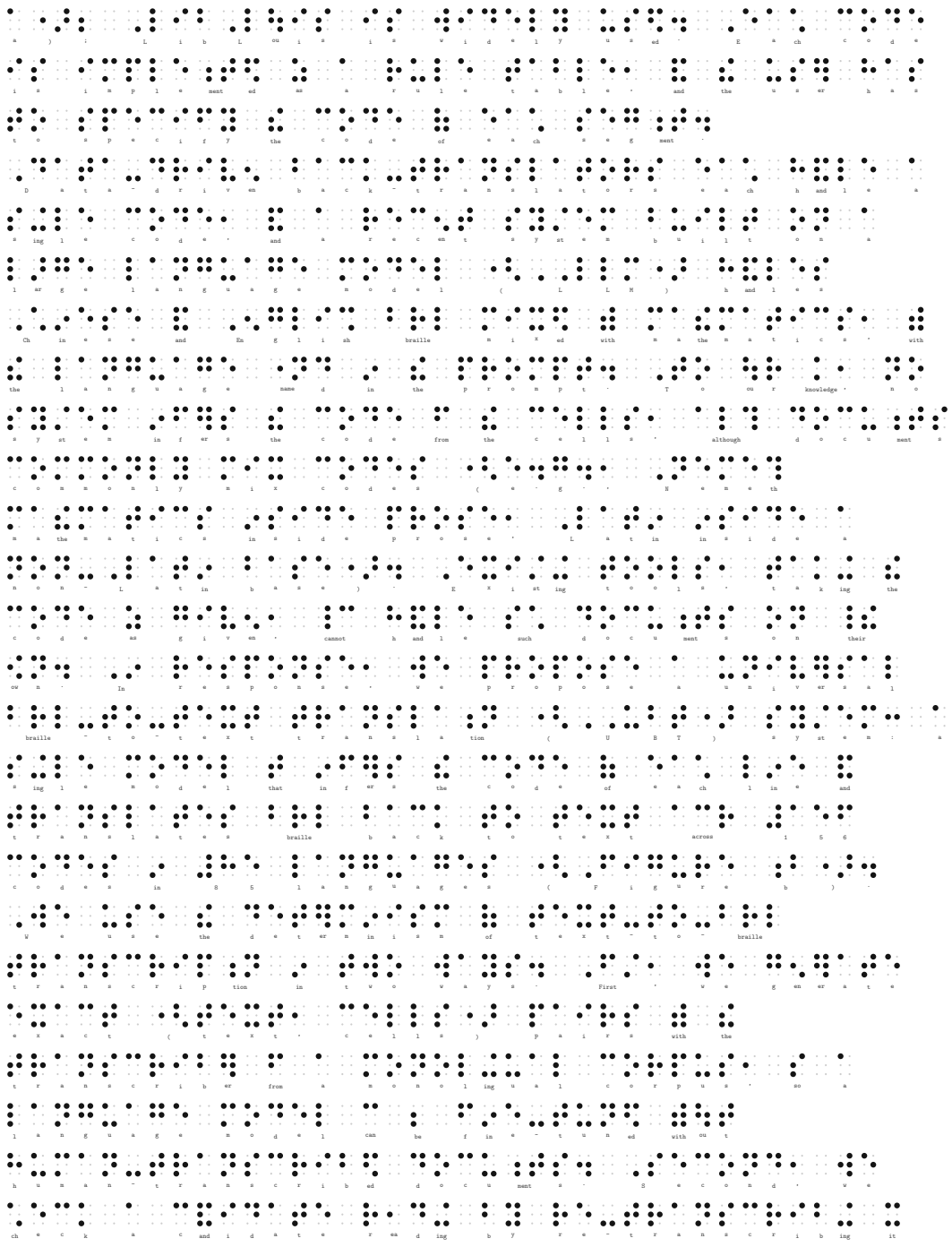
Figure 8: (continued)

units, 172 are verified and 172 of them are exact; at the cell level, 24 of 20,233 cells (0.12%) carry a character error. A sentence-level version of the same check (95 markup-free sentences from the main text, code given) behaves the same way: under best-of-50, 93 of 95 sentences are verified and all 93 are exact (0.03% CER against 0.14% for greedy), and re-running the unverified two at n=100 verifies none of them.

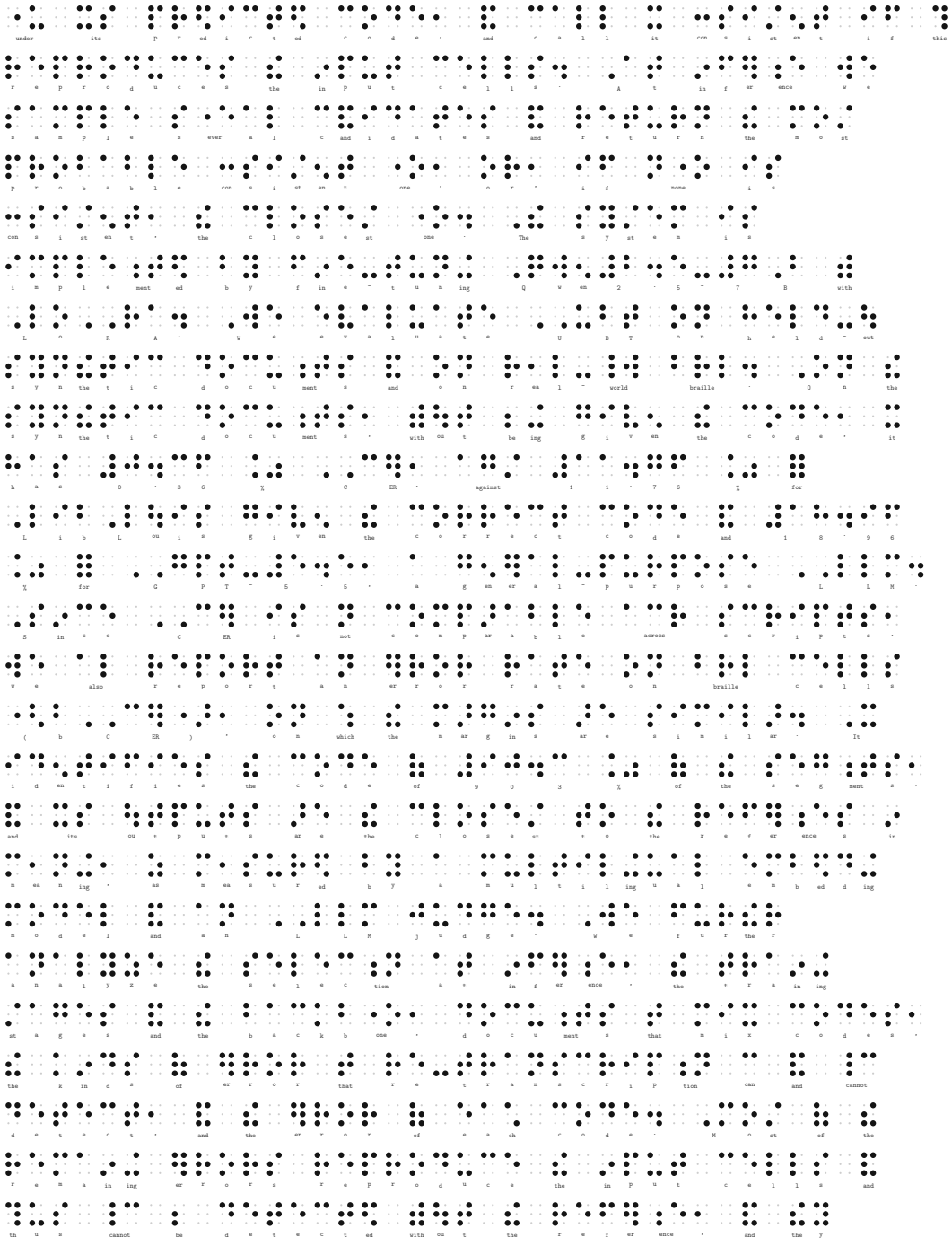
The pages that follow reproduce all 22 embossed pages, with the inkprint each run of cells encodes set beneath it. Cells shaded yellow belong to a unit the verifier declined to verify; cells shaded red were decoded wrongly, with their inkprint set in red. Every red cell lies inside a yellow span. The inconsistent region covers 855 cells (4.2% of the edition), since a single unrecovered character condemns its whole unit.

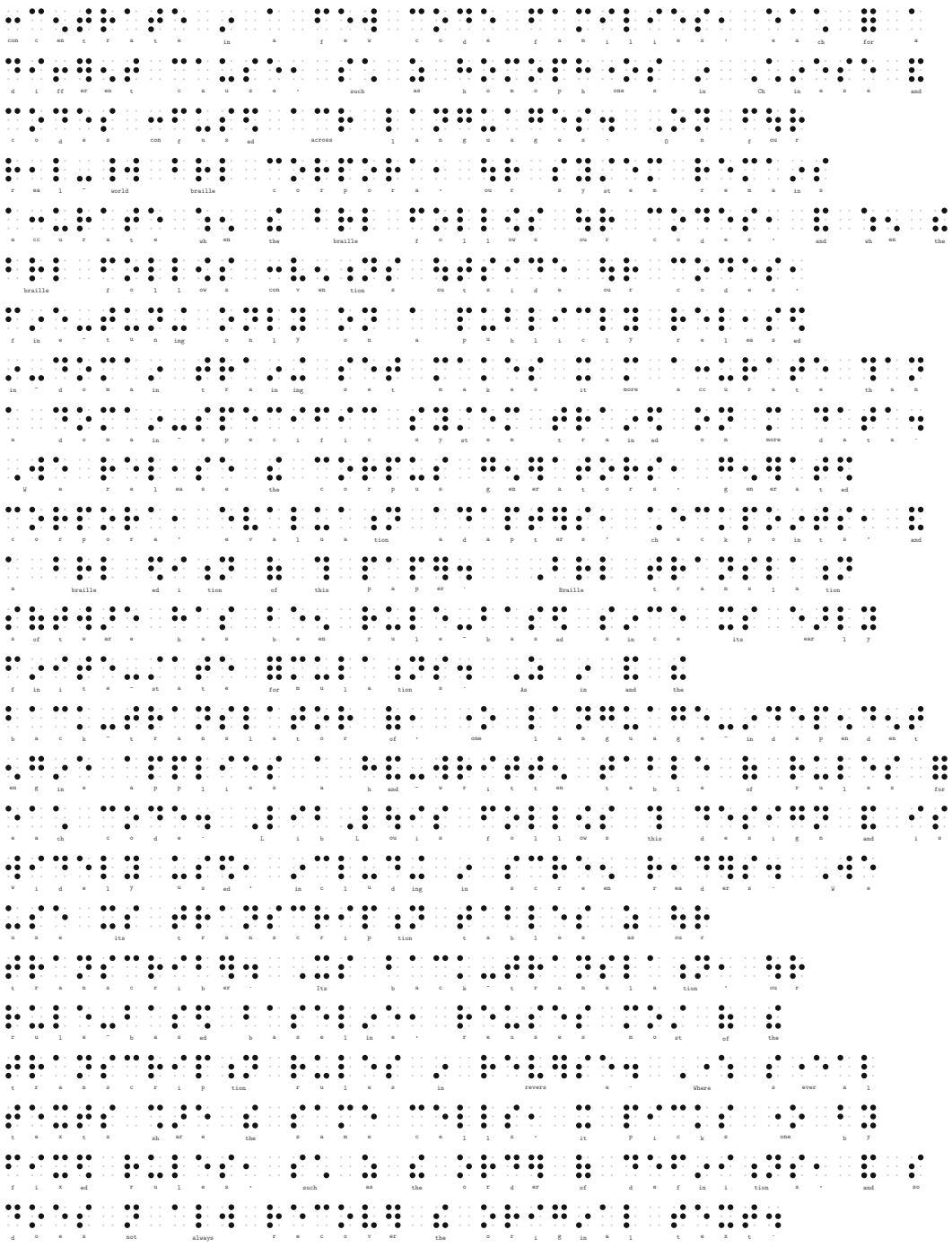


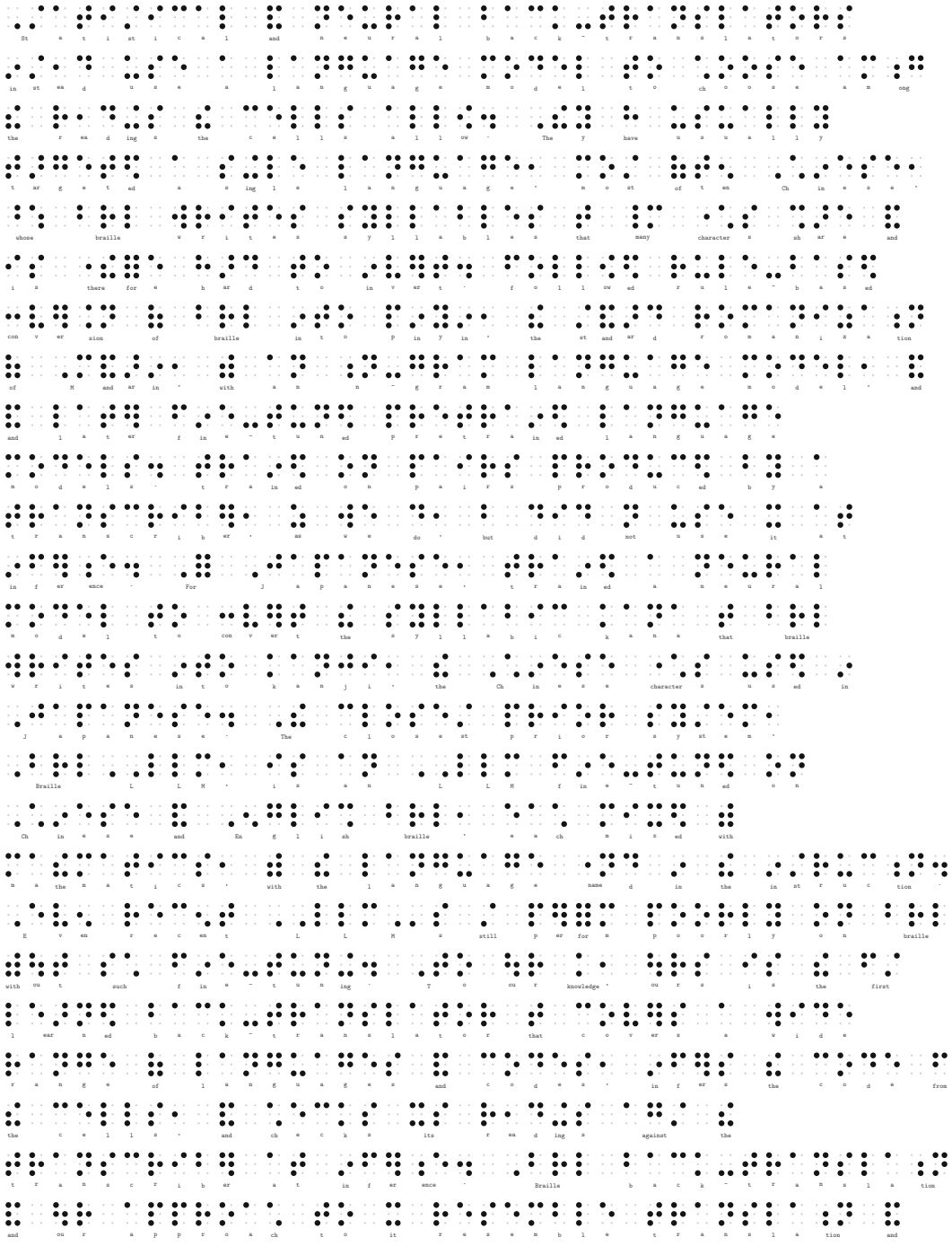
Braille edition, page 1 of 22



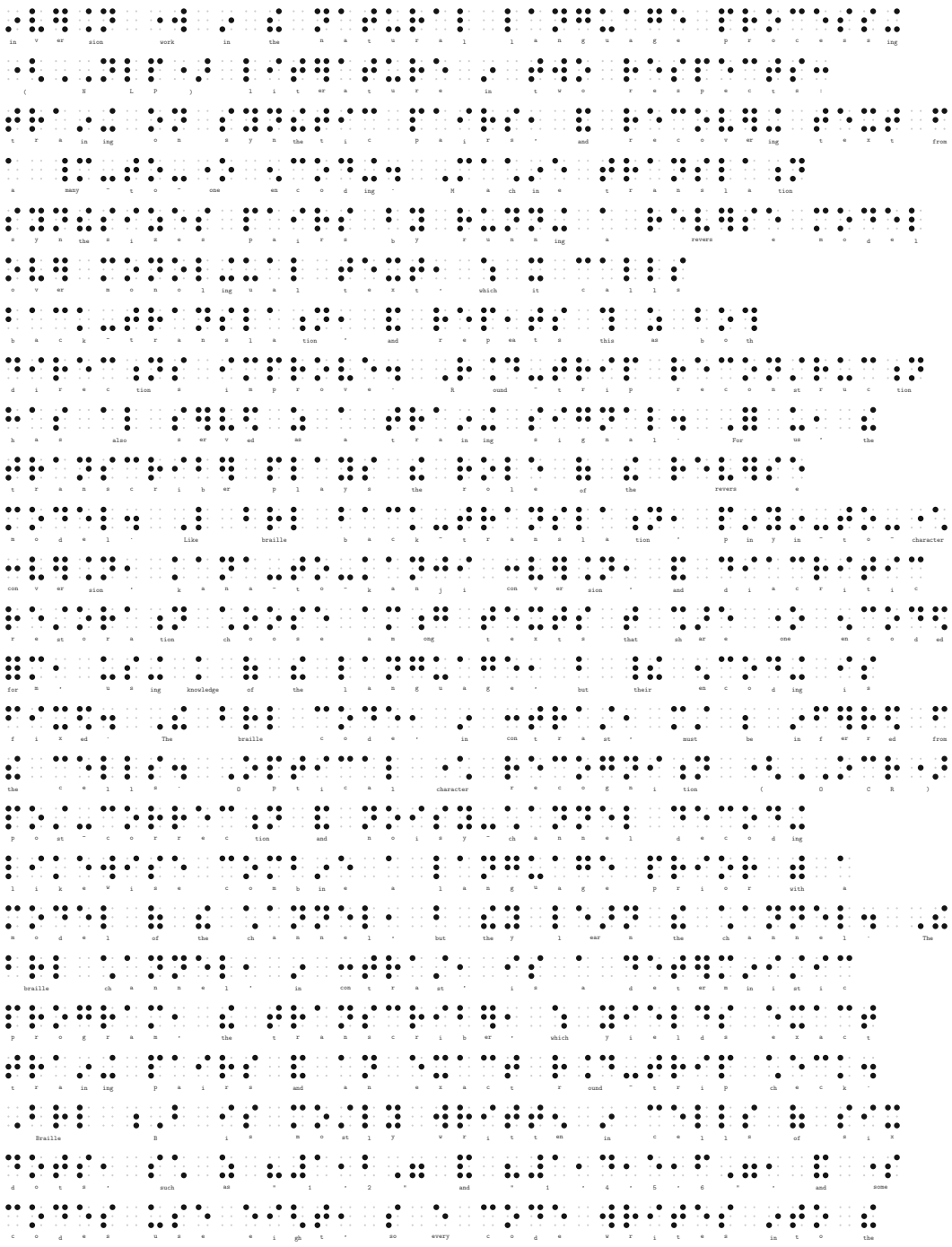
Braille edition, page 2 of 22


under its Prefixed code . and calling it consistent this
reproduces the input Cells . At interface none
sample ever all candidate sets and returns the most
probable consistent one . or if none is
consistent . the closest one The system is
implemented by fine-tuning Qwen2.5-7B with
LoRA . We evaluate on Braille-to-English
translation using synthetic data generated from
Project Gutenberg texts . Each sample pairs a
contracted Braille string with its uncontracted
English equivalent . The synthetic data
generation pipeline produces 10,000 training
pairs and 1,000 validation pairs . Each pair
consists of a randomly selected English sentence
and its Grade 2 Braille translation . The
translation is performed by liblouis with the
en-ueb-g2.ctb table . Sentences are drawn from
Project Gutenberg books and filtered to lengths
between 10 and 80 characters . Short sentences
under 10 characters are excluded because they
often contain proper nouns that are difficult to
translate without context . Long sentences over
80 characters are excluded to keep the maximum
sequence length manageable . The dataset is split
90% for training and 10% for validation . The
split is performed at the sentence level to
ensure no overlap between training and validation
data . Each sample is formatted as a chat with a
system prompt instructing the model to translate
Grade 2 Braille to English . The user message
contains the Braille string . The assistant
message contains the English translation . The
system prompt is identical for all samples . The
model is trained to produce only the English
translation with no additional text . Training
uses the AdamW optimizer with a learning rate of
2e-4 and a cosine schedule with 100 warmup steps
. The batch size is 4 with gradient accumulation
over 4 steps for an effective batch size of 16 .
Training runs for 1 epoch over the 10,000
samples . Gradient checkpointing is enabled to
reduce memory usage . The maximum sequence length
is 512 tokens . Evaluation is performed every 100
steps on the validation set . The primary metric
is character error rate . Character error rate is
the edit distance between the predicted and
reference strings divided by the reference length
. Word error rate is also reported . Word error
rate is the edit distance between the predicted
and reference word sequences divided by the
reference word count . Exact match accuracy is
the fraction of predictions that match the
reference exactly . Results are reported
separately for Grade 1 and Grade 2 Braille . The
model is also evaluated on a held-out set of
Braille produced by different translators than
the training data . This tests generalization to
different transcription styles . Error analysis
examines common failure modes . Typical errors
include misinterpreted capitalization indicators
and confusion between similar contractions . The
model sometimes drops or duplicates words in
long sentences . Performance degrades on
sentences containing numbers or mathematical
notation . Future work will address these
limitations with targeted data augmentation .
Braille edition, page 3 of 22

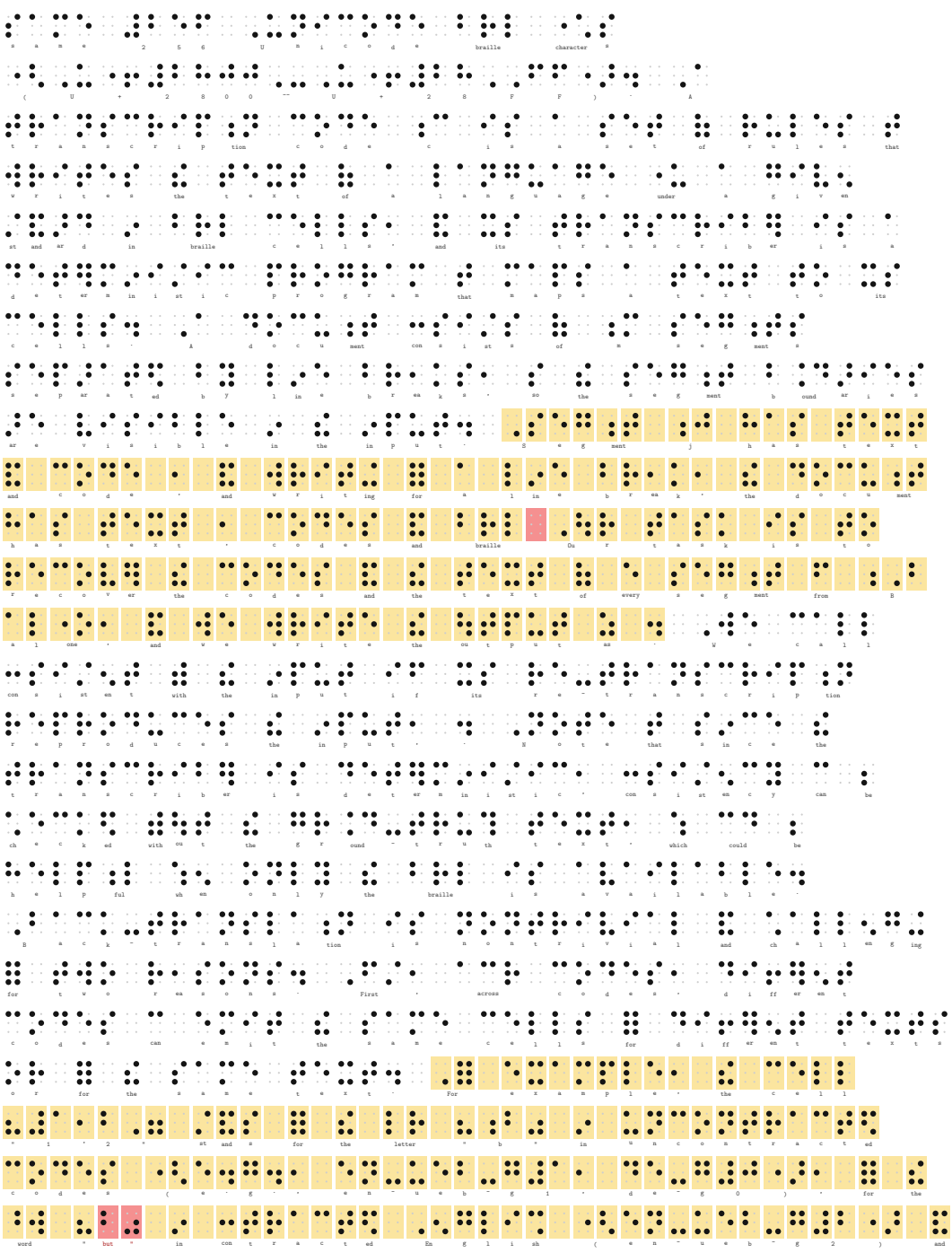




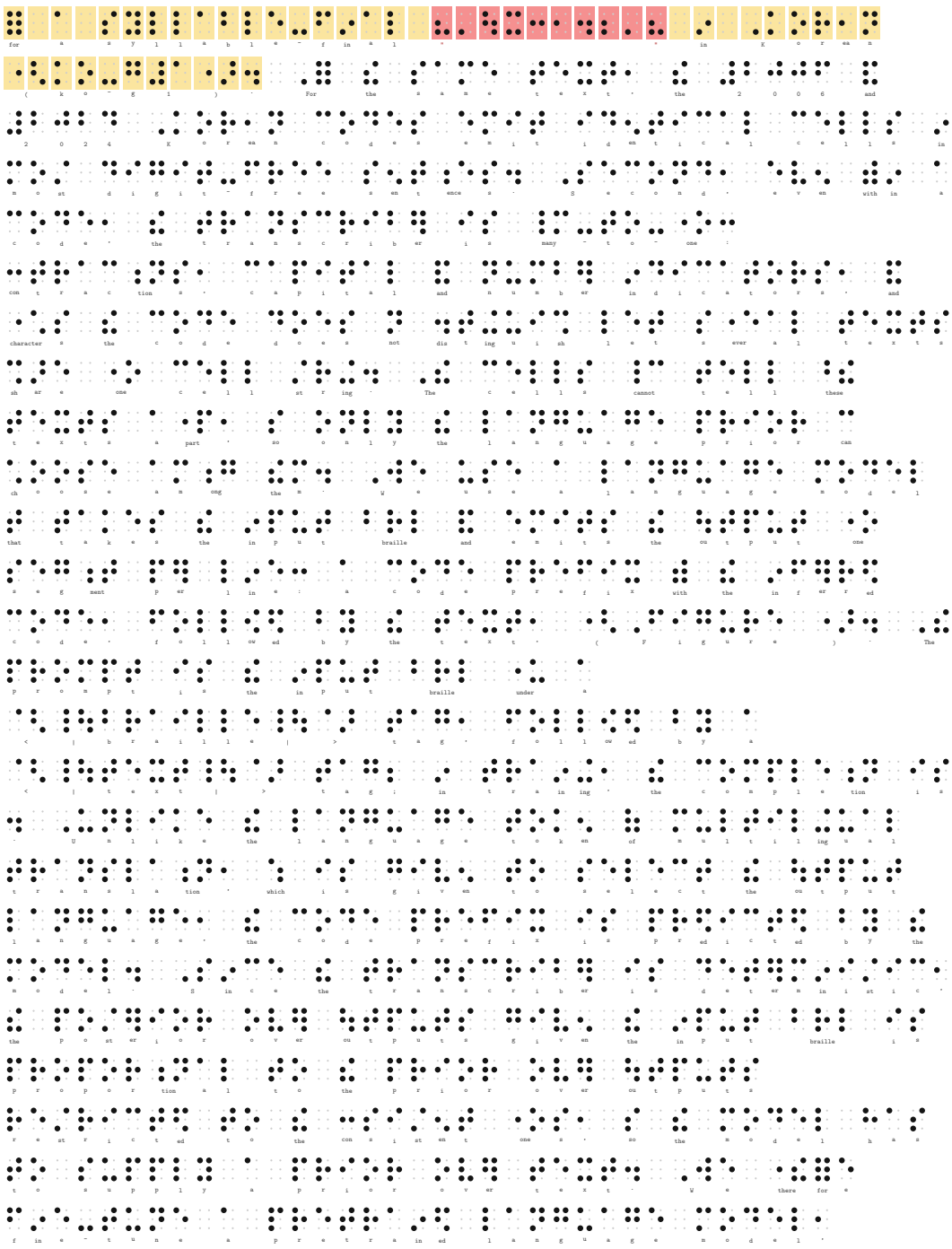
Braille edition, page 5 of 22

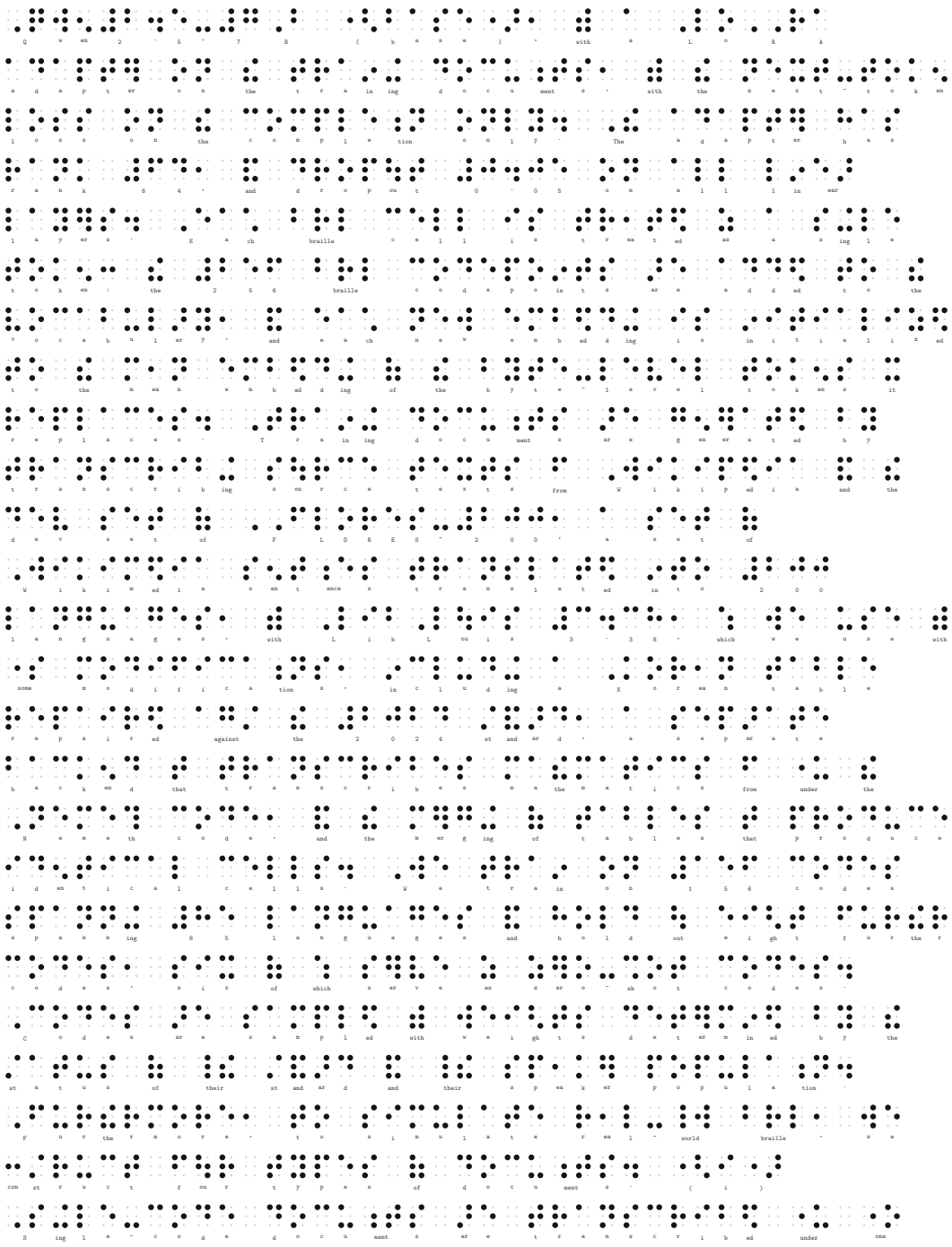


Braille edition, page 6 of 22

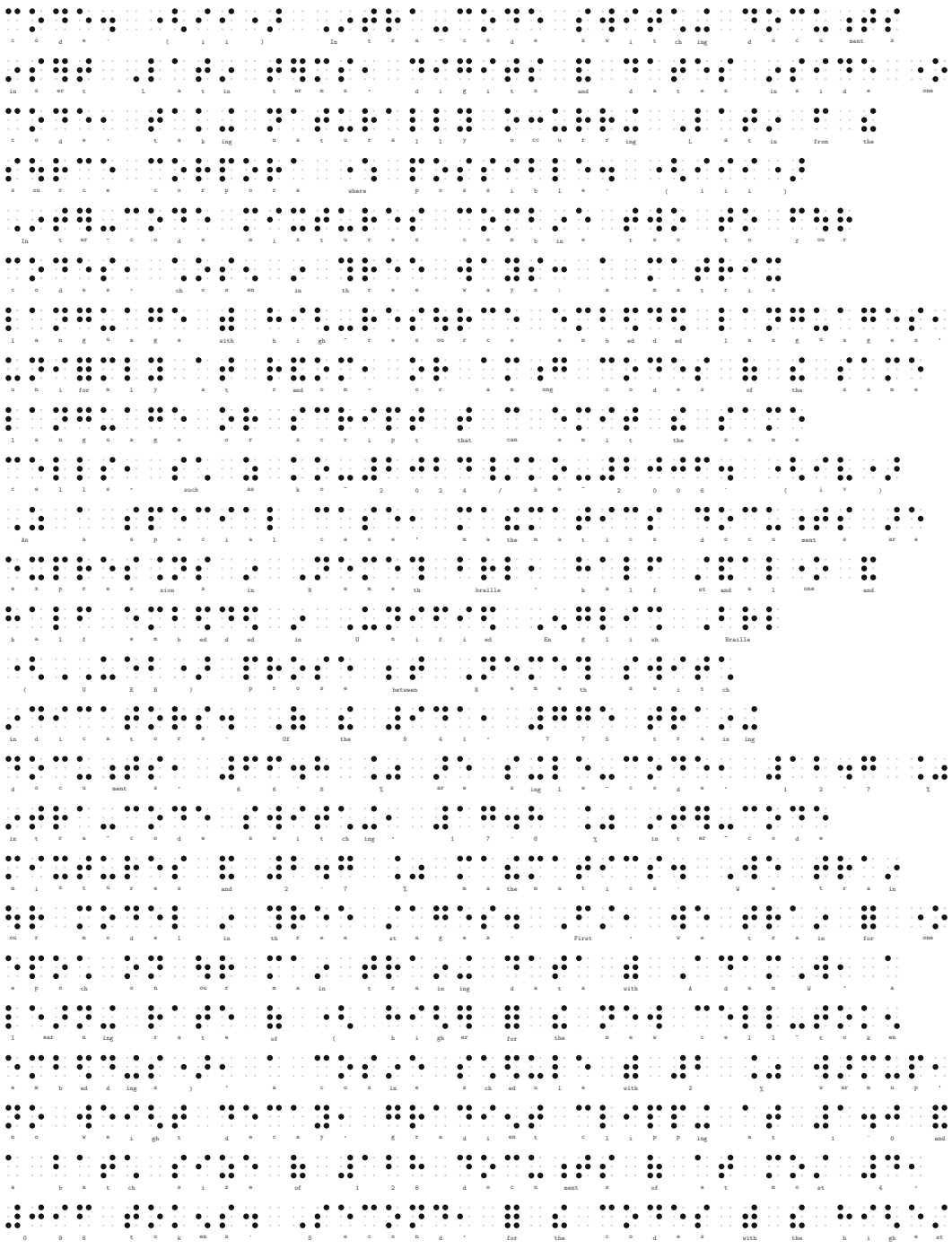


Braille edition, page 7 of 22

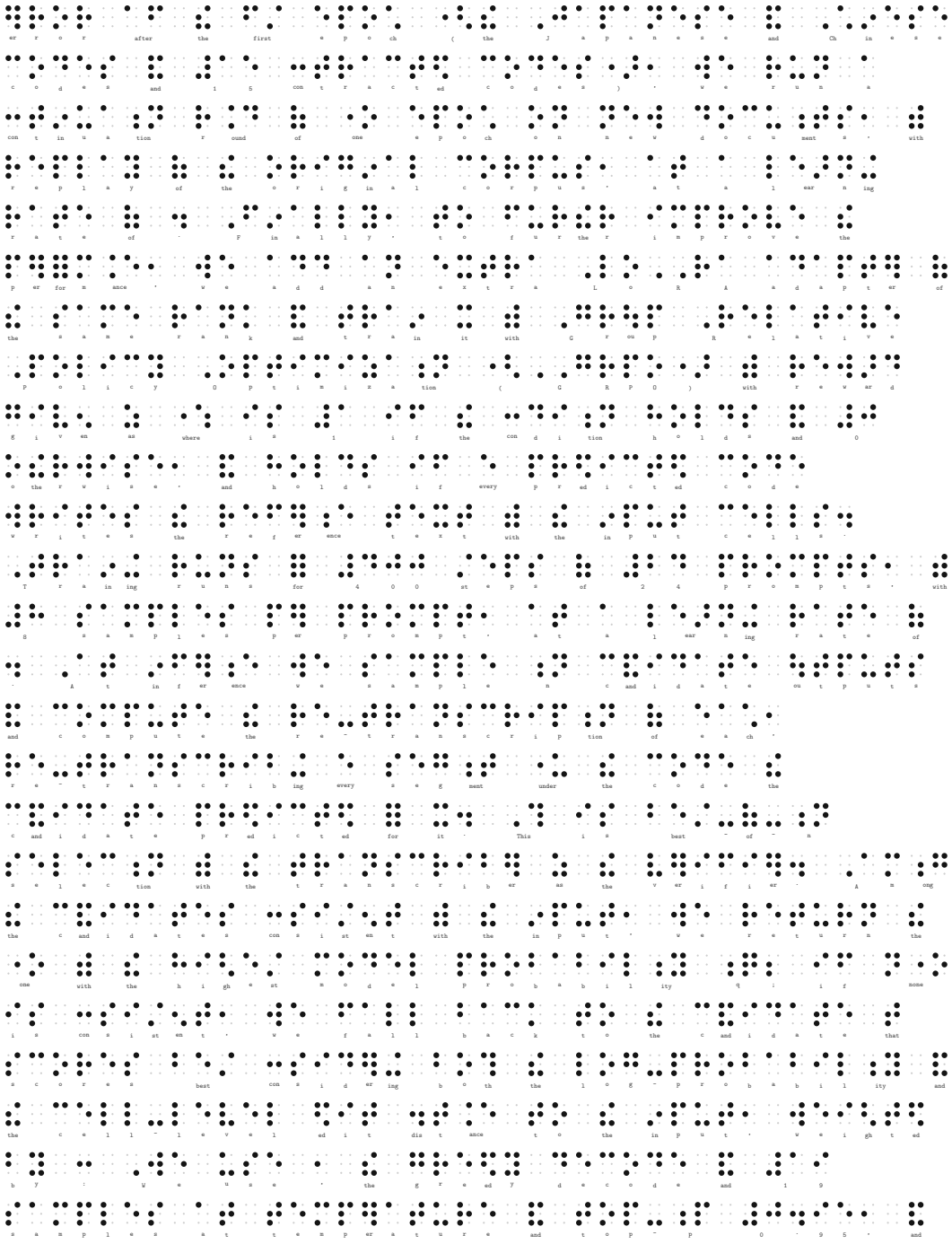




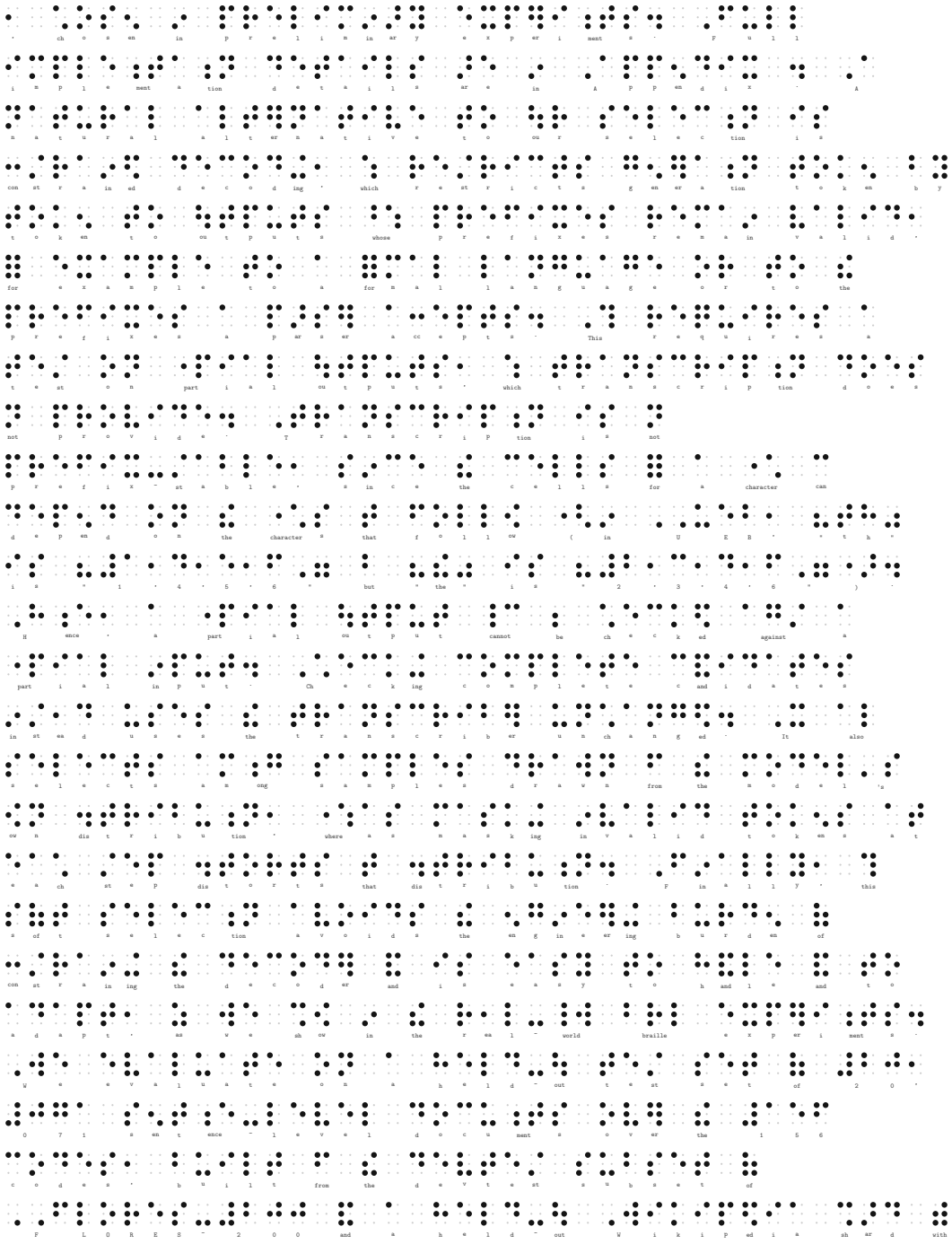
Braille edition, page 9 of 22



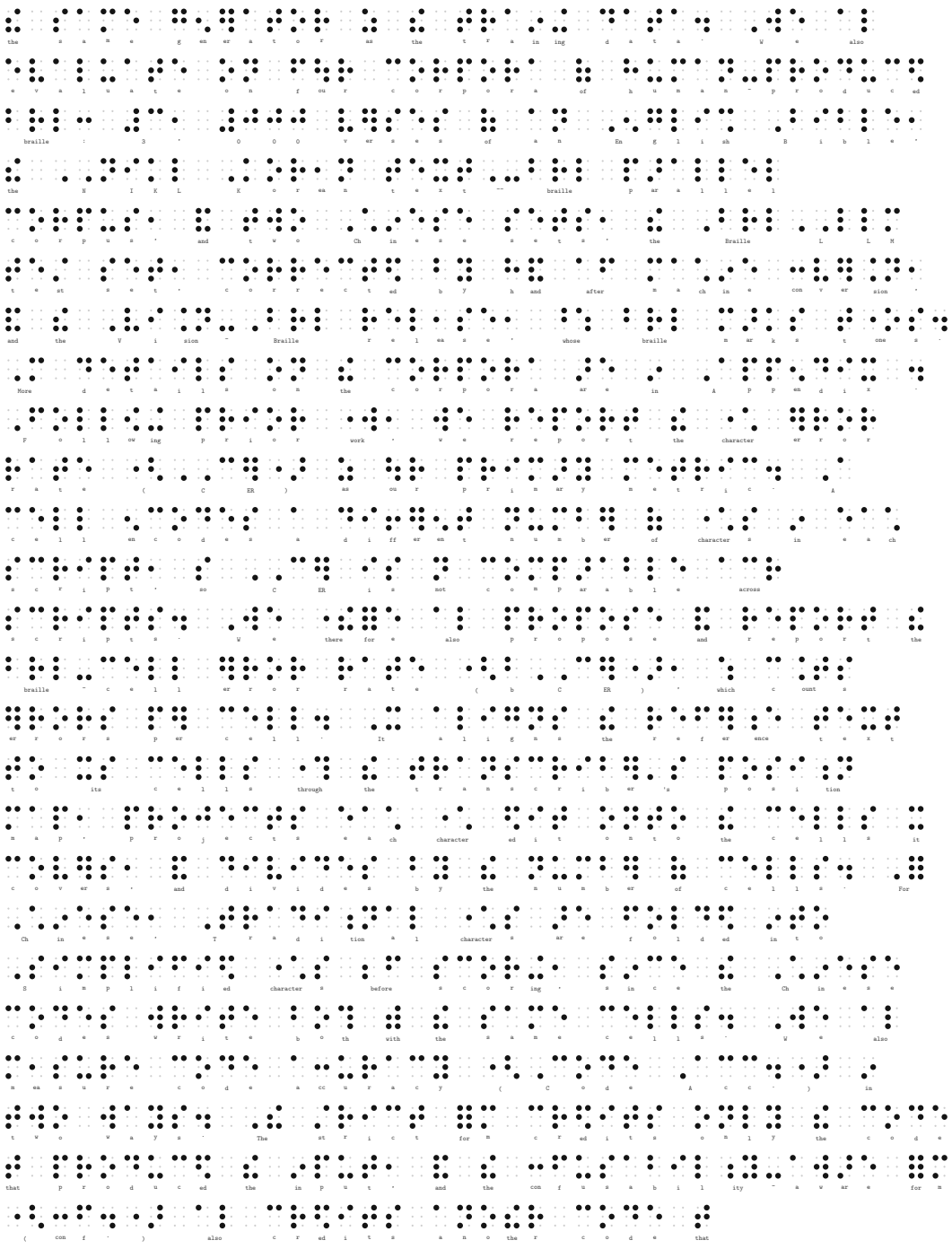
Braille edition, page 10 of 22



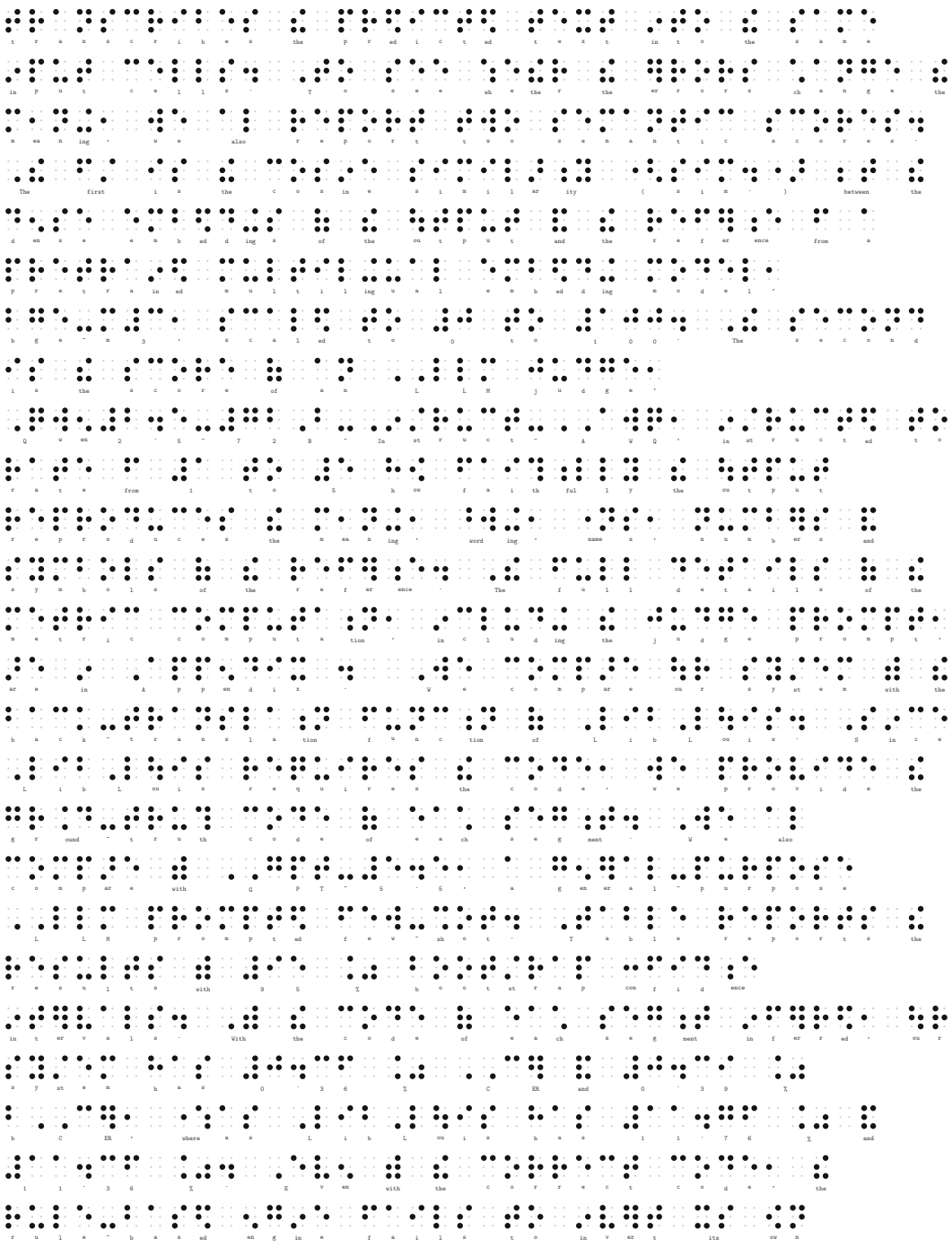
Braille edition, page 11 of 22



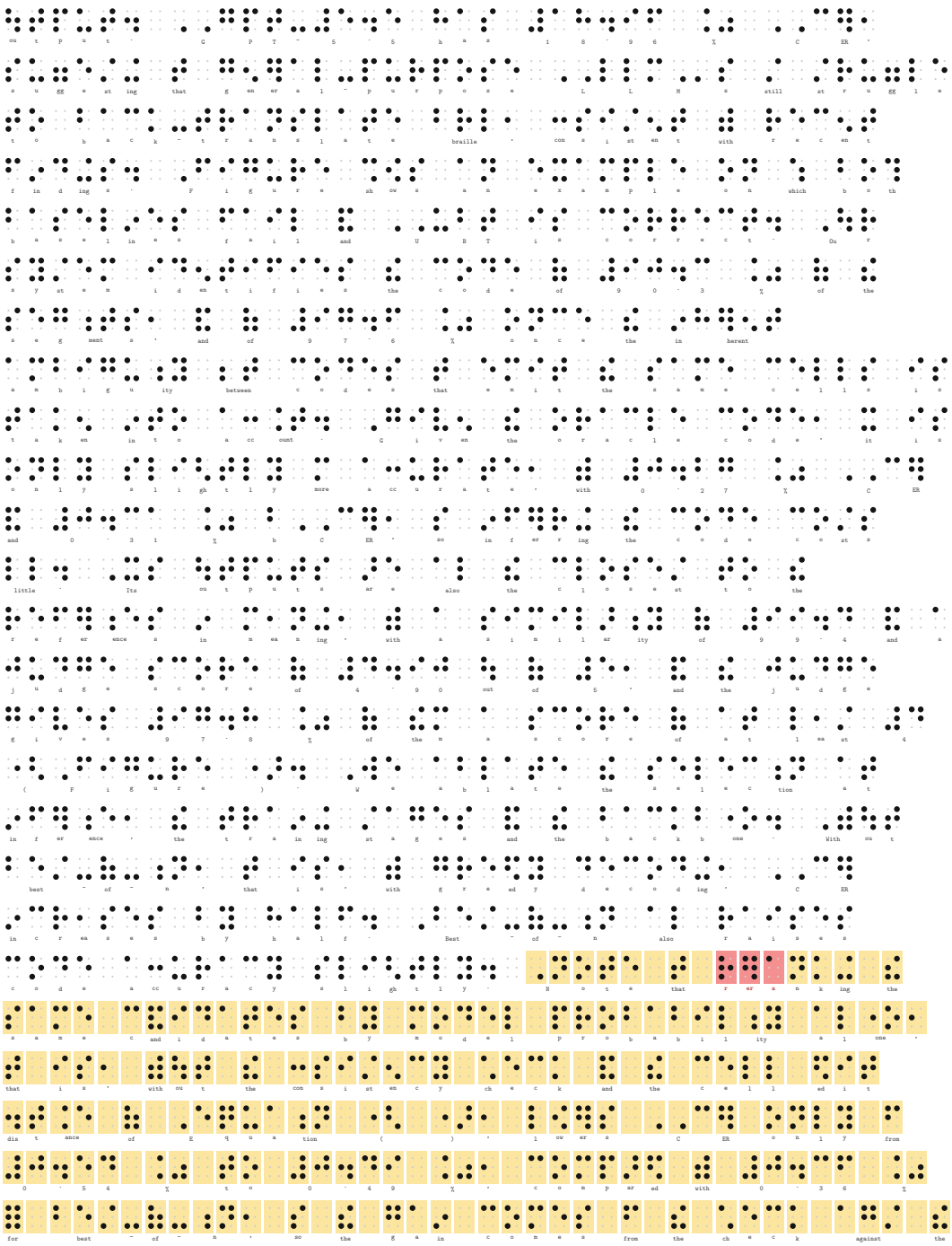
Braille edition, page 12 of 22



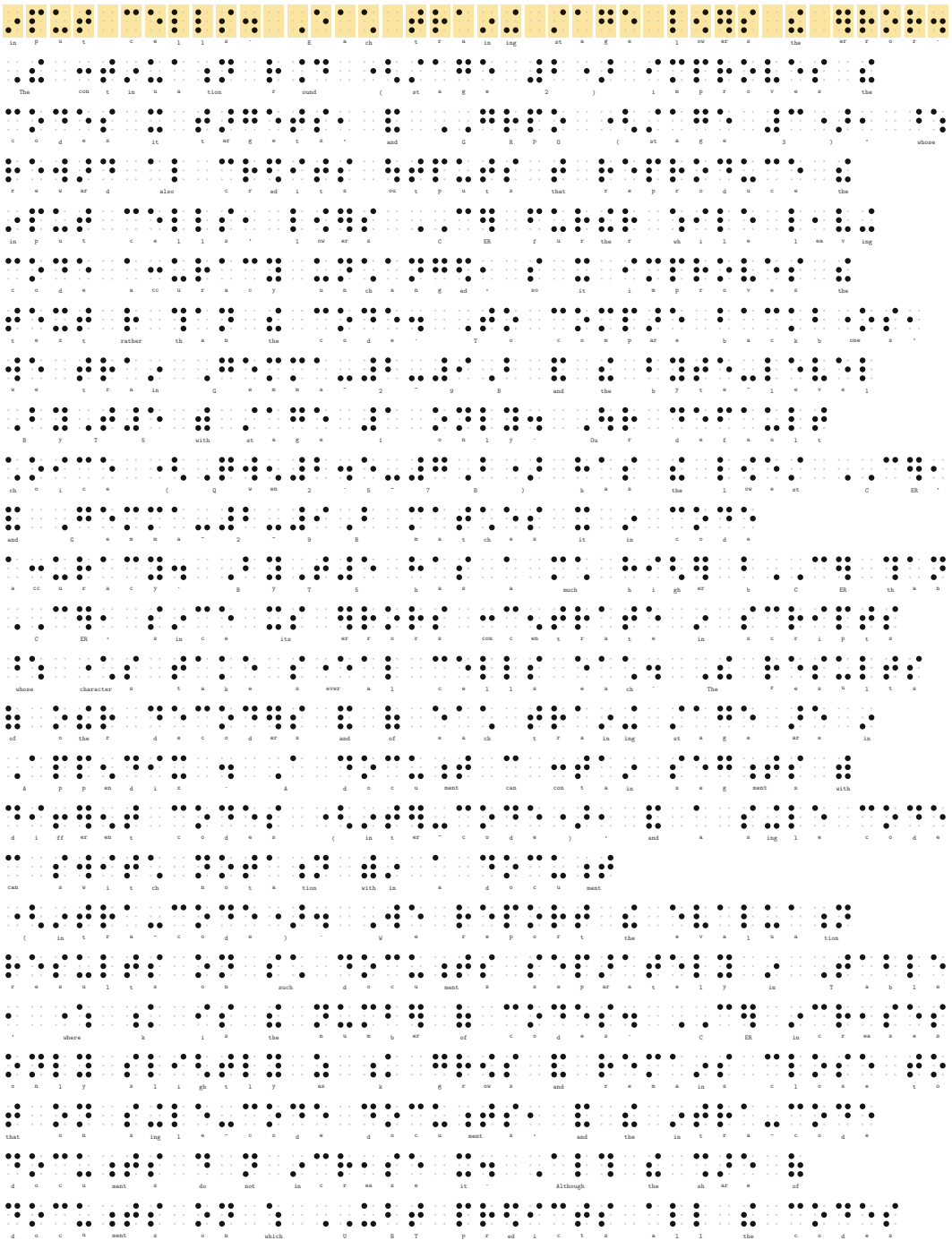
Braille edition, page 13 of 22



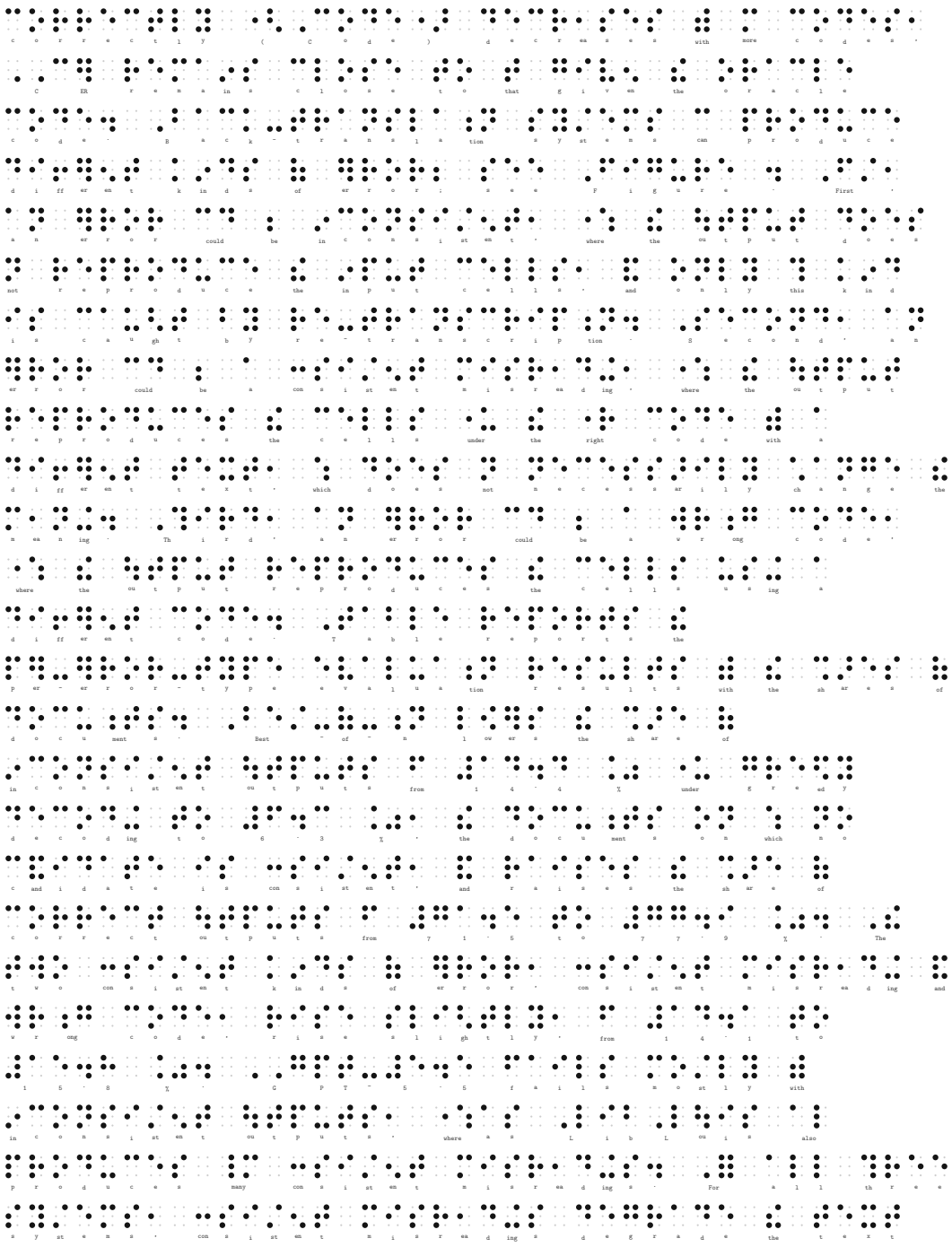
Braille edition, page 14 of 22



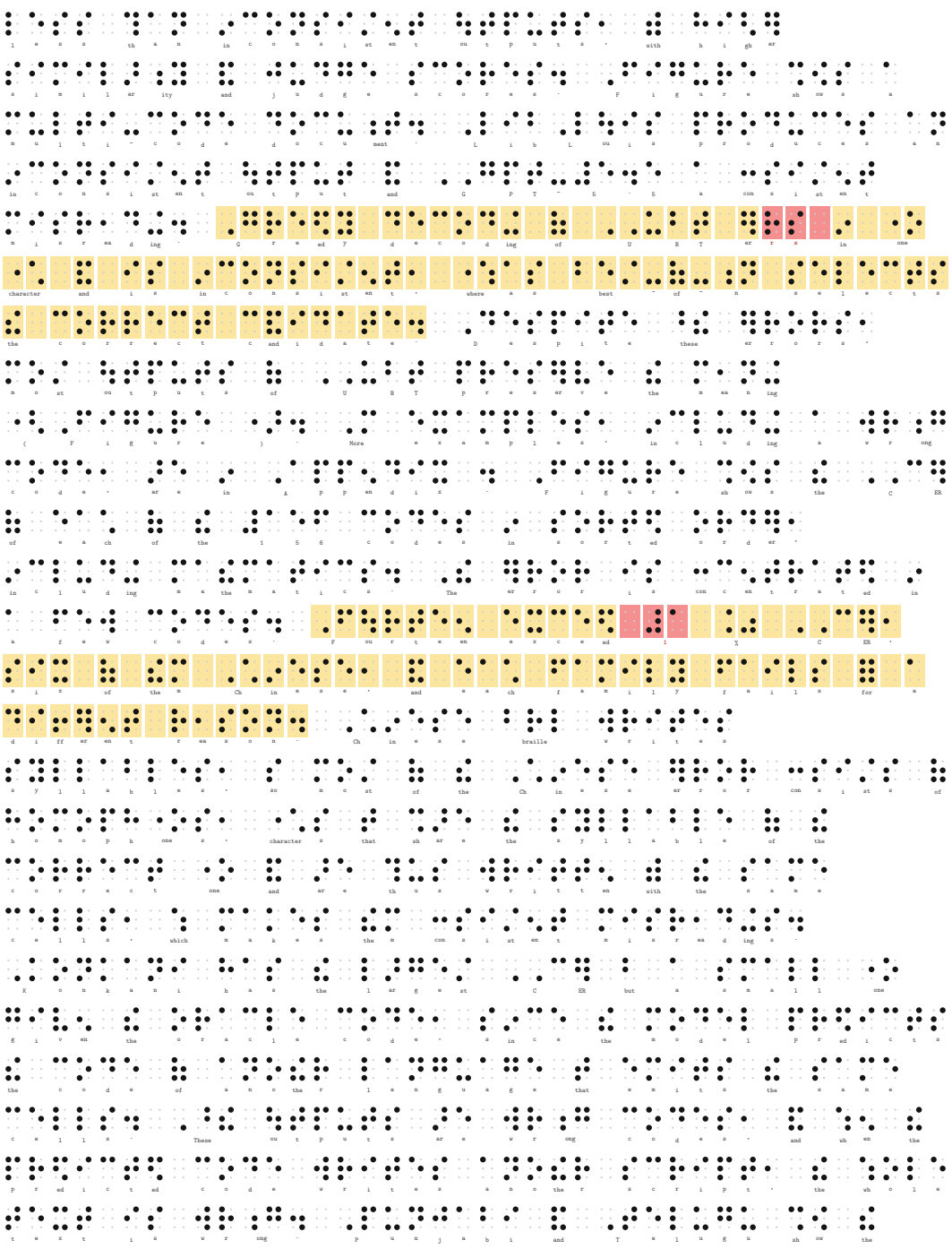
Braille edition, page 15 of 22

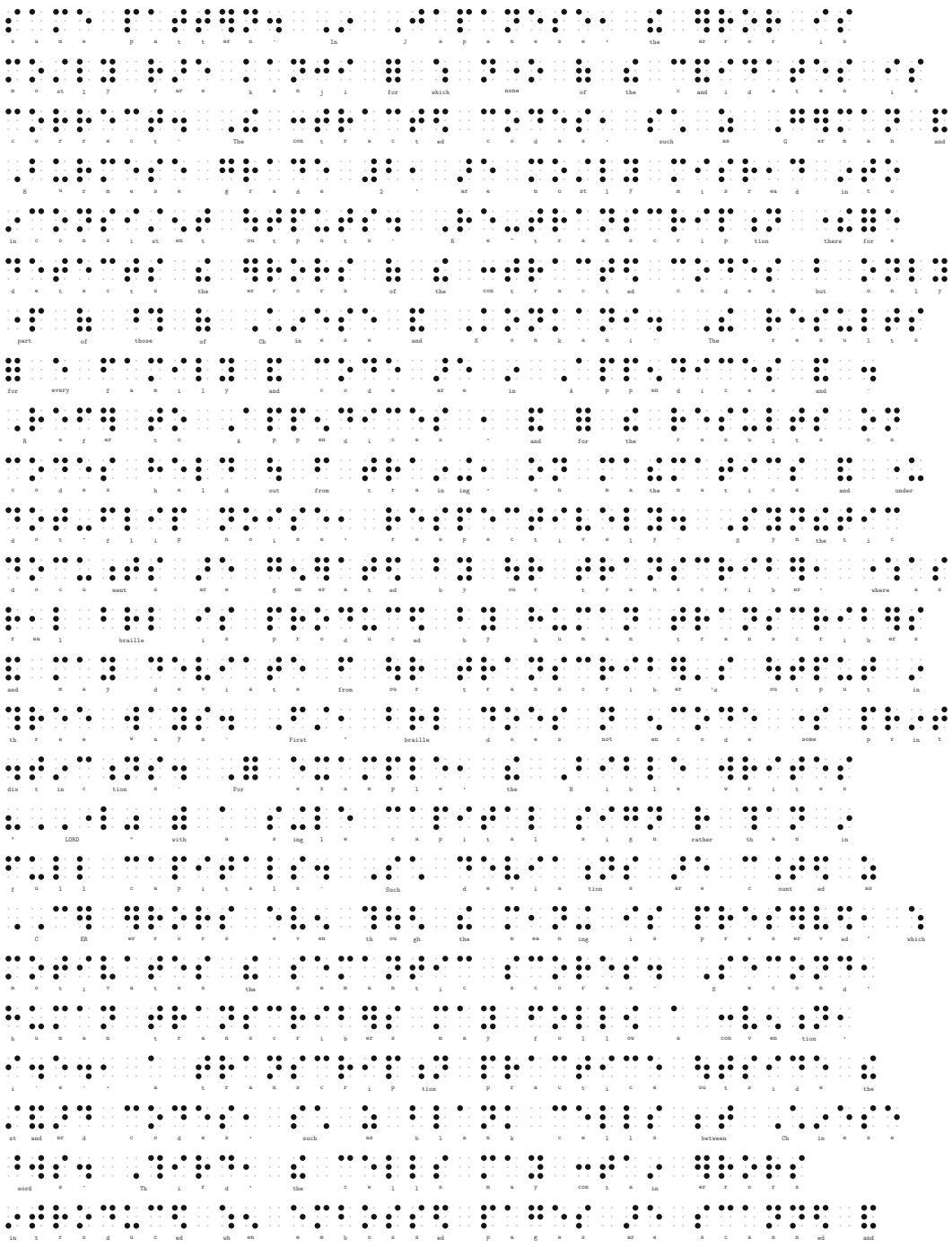


Braille edition, page 16 of 22

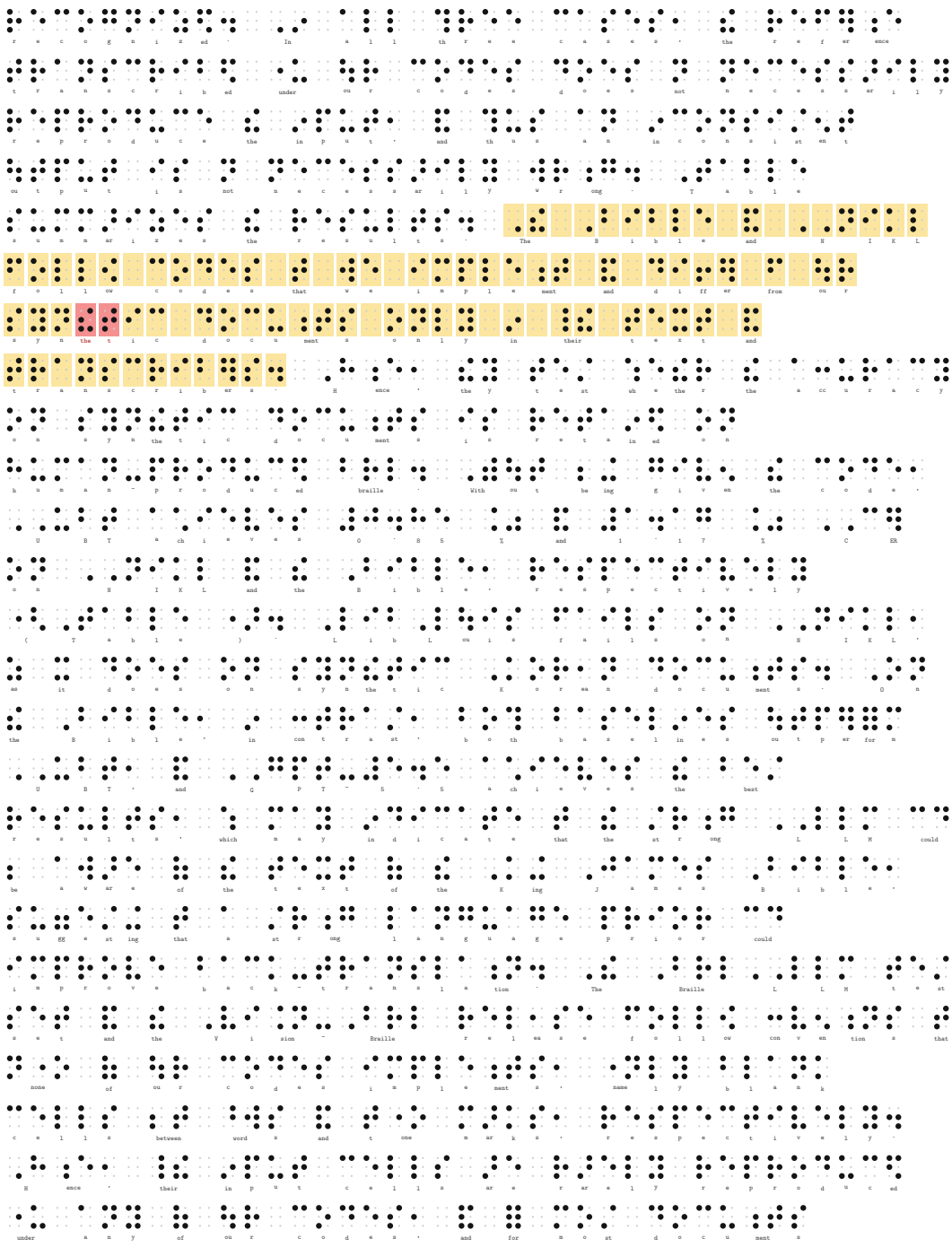


Braille edition, page 17 of 22

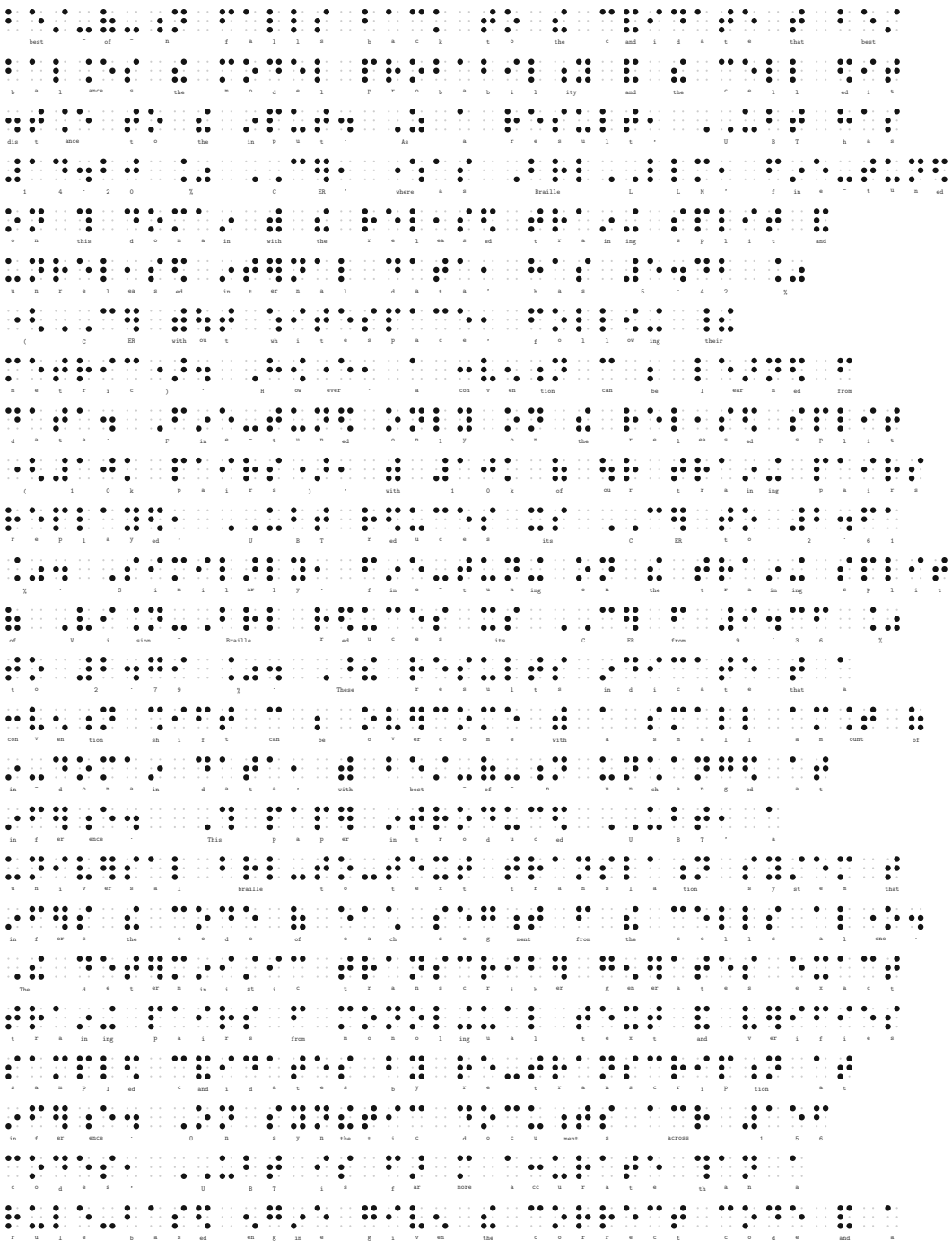




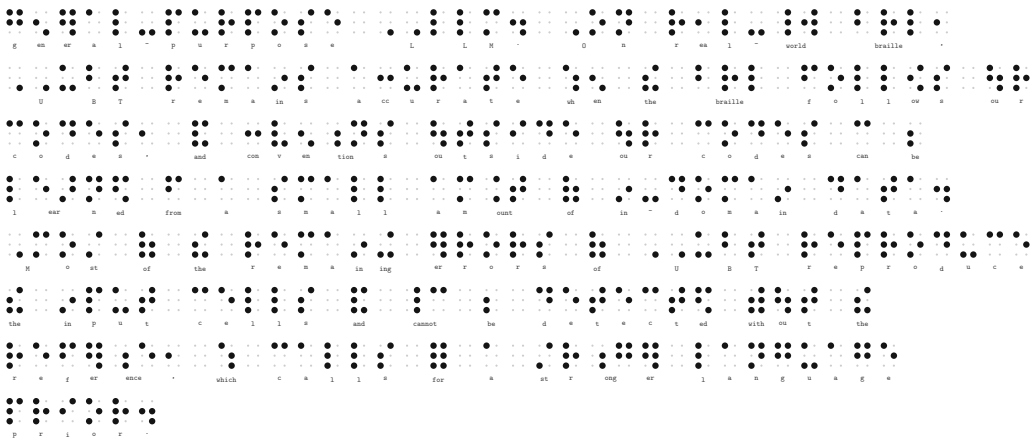
Braille edition, page 19 of 22



Braille edition, page 20 of 22



Braille edition, page 21 of 22



Braille edition, page 22 of 22