Language-model compression of famous text is mostly memorisation

Two small open models used as arithmetic coders on the Universal Declaration of Human Rights, a Gutenberg book and text written after their training, against xz and a PPM context model

Zain Dana Harper

Version 1.0, 9 November 2026. Finding F515 in the findings ledger. Signal (tier B).

Abstract

A language model that predicts the next word well is also a compressor, and its compression rate is often quoted as a measure of how well it models text. We used two small open models, SmolLM2-360M and Qwen3-0.6B, as arithmetic coders and compared them with xz and with a PPM context model. SmolLM2 encodes the English Universal Declaration of Human Rights at 0.27 bits per character, and Qwen3-0.6B at 0.57. The best classical coder, PPM trained on 5 million characters of Project Gutenberg, needs 1.99. We then added a control we had not planned: 30,000 characters of our own technical writing from October 2026, written after both models were released. On text written after their training, the two models need 1.59 and 1.66 bits per character. Cold PPM, which has seen nothing, finds the unseen text only slightly harder than the declaration: 2.70 against 2.57 bits per character. Most of the models' advantage on the declaration therefore comes from having seen it. An arithmetic encoder and decoder round-tripped a 120-token passage exactly with both models.

1 What this means for a reader

When a paper or a model card reports how few bits a language model needs per character, ask what text it was measured on. On a famous public document, the number measures recall of that document as much as modelling of language. On text the models could not have seen, their lead over a classical compressor shrinks from about tenfold to under twofold. Check a compression figure on fresh text before reading it as either.

2 Introduction

A model that assigns a probability to each next symbol can drive an arithmetic coder, which spends close to minus $\log_2 p$ bits on a symbol given probability p and lets a decoder with the same model recover the text exactly. The better the predictions, the shorter the code. Delétang and colleagues [1] used this equivalence to show that large language models are strong general-purpose compressors, and pointed out that counting the model's own size changes the picture. Huang and colleagues [2] found that compression on external text corpora tracks benchmark scores almost linearly across about thirty public models, and proposed compression as an evaluation measure.

Compression on public text has a known weakness. If the text was in the training data, a model can predict it from memorisation, recall of that text, and not from a general model of language. Li and colleagues [3] built an evaluation on this point: they measured compression on Wikipedia, news, code and arXiv text dated before and after each model's training cutoff and reported that the rate worsens after the cutoff for many models.

Our question is smaller and concrete. How many bits per character do two small open models need on a famous public document, against classical compressors, and how much of the result survives on text written after they were trained?

3 Preregistration and design

This test is one part of a larger study of language as a code. Its preregistration, a dated plan fixed and hash-locked before any result is seen, is a forecast file written and hashed on 1 November 2026 at 03:14 UTC, before any measurement. A first session wrote it and stopped with no measurement saved. A second session wrote a file of scoring rules 13 minutes later, before computing anything, and ran every measurement after that. A point forecast holds if the measured value is within 25 percent of it, and an inequality forecast holds when the inequality is true.

For this part the forecast file fixed:

- The bits per character on the English declaration: xz 3.0, PPM cold 2.4, PPM trained on Gutenberg 2.1, SmolLM2 1.2, Qwen3 1.0.
- On a held-out Gutenberg book: xz 2.9, trained PPM 2.1, SmolLM2 2.0.
- That SmolLM2 needs at least a billion characters to pay for its own size against xz, and trained PPM at most 200 million.
- A kill condition, a rule that decides against the run with no judgement after it: if an arithmetic-coded round trip with a model is not exact, the model rows count only as ideal lengths and not as real codes.
- Forecasts for the cost of sending known answers, and a shuffled-training control for PPM.

The unseen-text control was not in the forecast file. It was added after the declaration came in at 0.27 bits per character under SmolLM2, to test whether that number was memorisation. It is a post hoc control: it can test an explanation, and it carries less weight than a control fixed in advance.

4 Data and provenance

- The English Universal Declaration of Human Rights, 10,637 characters after runs of whitespace were collapsed, from the udhr2 collection of translations distributed with NLTK [7]. The download's source address was not recorded and the archive carries no licence file, so its licence is not established. We report numbers only and release no text from it.
- Lewis Carroll, Alice's Adventures in Wonderland, characters 5,000 to 35,000, from the Project Gutenberg selection distributed with NLTK [8]. Public domain in the United States.
- The training text for PPM: the first 5,000,000 characters of the other books in that selection, with Alice and the King James Bible left out.
- The unseen text: 30,000 characters of an unpublished research log of ours, written in October 2026. It is not released. Its SHA-256 is 2248f9fa40d14608bc9b2932af537caa0b548722e16499532413ca020952850c, so a later release can be checked against it.
- SmolLM2-360M [4], Hub revision f8027fd0, and Qwen3-0.6B [5], Hub revision c1899de2, both Apache-2.0, downloaded from the Hugging Face Hub on 31 October 2026 as safetensors with no remote code. The SmolLM2 weight file is 723,674,912 bytes and the Qwen3 file 1,503,300,328 bytes.

SmolLM2's technical report appeared in February 2025 and Qwen3's in May 2025 [4] [5]. That the unseen text was written after both models were trained rests on those dates and on the log's own dates.

5 Methods

Code length under a language model

Each text is split into tokens, the word pieces a model reads, with the model's own tokenizer. Every token t_i is priced at minus $\log_2$ of the probability the model gave it after the tokens before it. The ideal code length of a text of C characters and T tokens is

$$b = \frac{1}{C} \sum_{i} - \log_2 p(t_i \mid t_1, \ldots, t_{i-1}) \tag{1}$$

in bits per character, with the sum over tokens 2 to T. The first token is priced at $\log_2 V$ bits for a vocabulary of V tokens: 49,152 for SmolLM2 and 151,936 for Qwen3. Long texts are read in windows of 512 tokens that advance by 256, so every token after the first window is predicted from at least 256 tokens of context. The models ran in half precision on one GPU. Decoding the tokens gave back the exact original text for the declaration and Alice under both tokenizers.

A real code

An ideal length is only a claim until a coder achieves it. For each model, an arithmetic encoder turned the first 120 tokens of the declaration into a bit string, using a 64-token context, single precision weights on the CPU and probabilities rounded to integer frequencies out of 2^{24}. The decoder, given the bit string, the first token and the same model, rebuilt the 120 tokens. This is the kill condition in the design.

Classical coders

xz is a general-purpose compressor that replaces repeated strings with references to earlier copies; it ran at its strongest preset, and its size in bits is divided by the number of characters. PPM, prediction by partial matching [6], predicts each character from counts of what followed the same preceding characters, here up to 5 characters back, with method C escapes and exclusion. A character never seen in any context costs $\log_2$ of the number of Unicode code points. PPM cold starts empty and learns as it codes. PPM trained starts from counts over the 5,000,000 training characters and keeps learning as it codes; the counts are restored after each text.

Payback length

A compressor that must be sent along with the message pays for its own size. The payback length is the number of characters after which the bits saved over xz equal the model's size in bits:

$$N = \frac{8S}{b_{xz} - b_m} \tag{2}$$

where S is the model's size in bytes and b_{xz} and b_m are bits per character under xz and under the model. For trained PPM, S is the training text compressed with xz, 1,458,524 bytes.

Cost of a known answer

For a prompt such as "The capital of Norway is" or "5 + 6 =", the cost of an answer is the code length of the answer's tokens given the prompt. Each prompt has a right answer and a wrong one. There are 20 capital-city prompts and 30 each of one-digit and three-digit sums.

6 Controls and planted checks

A control is a version of the data with the structure under test removed and scored the same way. Three serve here.

The round trip of the methods section is a check that the model rows are real codes. An arithmetic encoder and decoder round-tripped a 120-token passage exactly with both models. The kill condition did not fire. The long-document figures remain ideal code lengths.

PPM was retrained on the same 5,000,000 characters with every character shuffled, and again with every word shuffled. On the declaration, character-shuffled training gave 3.59 bits per character, worse than cold PPM at 2.57, so the trained model's gain over cold is gone and more. Word-shuffled training gave 2.10 against 1.99 for normal training, a loss of a fifth of the gain. The context model gains from letter order inside words much more

than from word order, as it should.

The unseen text is the control for prior exposure. A coder that has seen nothing, cold PPM, gives a measure of how hard each text is in itself.

7 Results

Table 1 gives every coder on every text, and Figure 1 draws the same numbers.

Table 1. Bits per character for each coder and text. Lower is better. The unseen text is a post hoc control.

Text	xz	PPM cold	PPM trained	SmolLM2-360M	Qwen3-0.6B
English declaration, 10,637 characters	2.83	2.57	1.99	0.27	0.57
Alice, Gutenberg, 30,000 characters	3.12	2.84	2.14	0.89	1.16
Unseen, October 2026, 30,000 characters	2.74	2.70	2.89	1.59	1.66

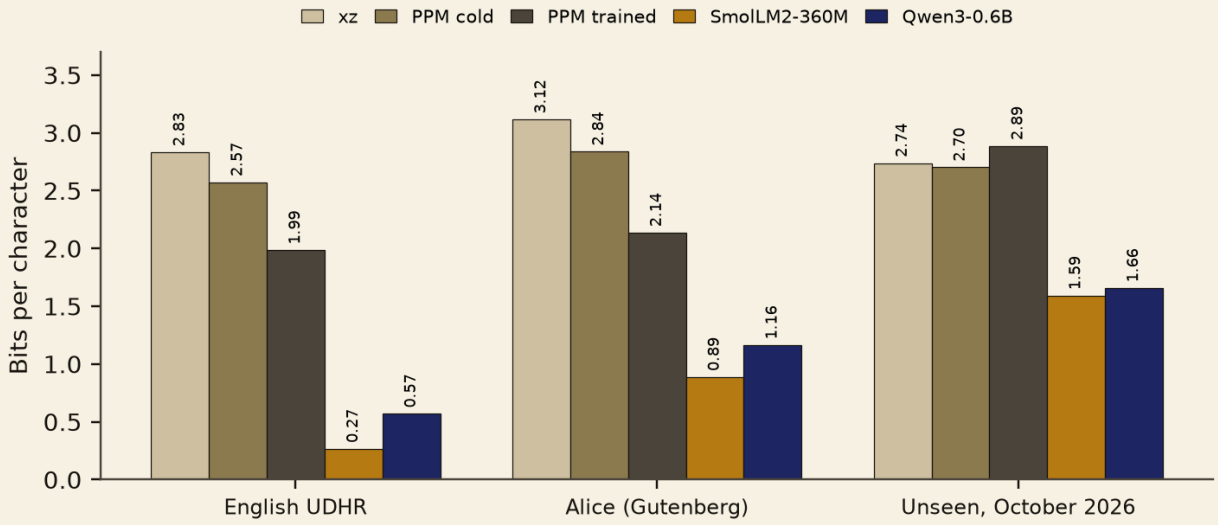


Figure 1. Bits per character for each coder on the three texts. The language models' lead over the classical coders is largest on the declaration and smallest on the unseen text. Drawn from the study's result files.

SmolLM2 encodes the English Universal Declaration of Human Rights at 0.27 bits per character, and Qwen3-0.6B at 0.57. Against xz on the same text that is a tenfold and a fivefold lead. Our forecasts were 1.2 and 1.0, and both failed by far more than the 25 percent allowed. The Gutenberg book shows the same pattern more weakly: 0.89 under SmolLM2, against a forecast of 2.0.

On text written after their training, the two models need 1.59 and 1.66 bits per character. SmolLM2 needs about six times the bits it needs on the declaration. Its lead over xz falls from tenfold to under twofold.

Cold PPM, which has seen nothing, finds the unseen text only slightly harder than the declaration: 2.70 against 2.57 bits per character. xz finds it easier, 2.74 against 2.83. A sixfold jump for a language model on a text that classical coders find about as hard as the declaration points at prior exposure to the declaration.

Trained PPM does worse than cold PPM on the unseen text, 2.89 against 2.70 bits per character. Its training on literary English helps on the declaration and on Alice and hurts on our technical writing.

Model size changes the ranking for short messages. By eq. 2, SmolLM2 pays for its 723,674,912 bytes against xz after 2.6 billion characters of Alice-like text, which held the preregistered bound of at least one billion. On

the unseen text, where its lead over xz is 1.15 bits per character, the same arithmetic gives about 5 billion characters. Trained PPM pays back after 11.9 million characters of Alice-like text.

Figure 2 shows the cost of sending a known answer. A right capital city costs about 1 bit under either model, and a wrong one 18 to 20. A one-digit sum costs Qwen3 1.29 bits and SmolLM2 4.01, close to its 4.63 for a wrong sum. A three-digit sum costs 8.5 to 9.1 bits even when right, and a wrong one costs only about 5 bits more. The answer costs fit recall of capital cities and show no sign of reliable three-digit addition.

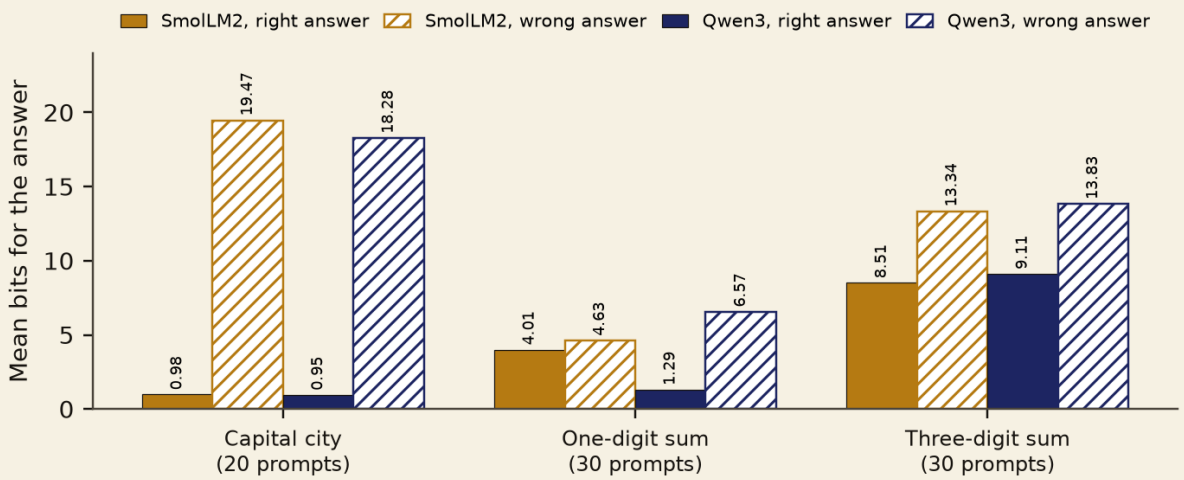


Figure 2. Mean code length of a right and a wrong answer given its prompt, for each model and task. Drawn from the study's result file.

Appendix A sets every preregistered forecast for this part against its outcome.

8 Discussion

The headline number, 0.27 bits per character, is true of the declaration and says little about language. The declaration is copied widely on the public web, so we infer that it sits in the training data of both models, probably many times; we did not check their training sets. A model that has seen it often can code it from recall. The unseen text removes that route, and the lead over classical coders drops to what a model of language buys.

That residual lead is still real. On text neither model could have seen, both need about 1.6 bits per character against 2.70 to 2.89 for the classical coders. This agrees with the main claim of Delétang and colleagues [1] that language models are strong compressors, and with their point that the model's size belongs in the count: on fresh text, SmolLM2 needs billions of characters before it pays for itself against xz.

Our result fits the evaluation design of Li and colleagues [3], who score compression on text from after each model's training cutoff. It also bears on compression used as a measure of capability [2]. A corpus that overlaps the training data rewards recall, and two models can rank differently on recall than on language. Here the smaller model, SmolLM2, leads by a factor of two on the declaration and by 4 percent on the unseen text.

The trained PPM result is a smaller lesson of the same kind. Counts from Gutenberg books help on literary text and hurt on a technical log. A compressor's training text is part of its result, for a context model as much as for a neural one.

9 Limitations

- There is one unseen text. It is our own technical writing in one register, full of numbers and technical terms,

and it was chosen after the headline numbers were seen. A set of fresh texts across registers is needed before the size of the memorisation share can be stated.

- The unseen text differs from the declaration in style as well as in novelty. Cold PPM finds the two about equally hard, which suggests the difficulty gap is small, but it does not remove the difference.
- That the unseen text was written after the models were trained rests on its date. We have not checked whether it quotes public text.
- The unseen text is not released, so no one else can rerun that row. Its hash is given so a later release can be checked.
- Both models are small, 360 million and 600 million parameters. Larger models may memorise more.
- The round trip covered 120 tokens. The long-document figures are ideal code lengths, and a real coder would add a few bits.
- All declaration and Alice rows for the language models carry the same contamination.
- The licence of the declaration file is not established; we report numbers only.
- Scoring was done by our own code and was not blind.

10 Verified

Verified: on 9 November 2026 every bits-per-character value in Table 1, the character counts 10,637 and 30,000, the token counts, the vocabularies, both model file sizes, the round trip on 120 tokens for both models, the answer costs, the shuffled-training values, the trained PPM model size, both preregistered payback values and every forecast in Appendix A were re-read from the study's result files and its score file. They match the findings ledger. The forecast file's SHA-256 was recomputed and matches the hash recorded beside it and in the study's lock. The unseen-text SHA-256 is the one recorded in its result file. The payback of about 5 billion characters on the unseen text is our arithmetic by eq. 2 from the recorded model size and the two rates. Both figures were drawn by a script that reads only the result files. The study was not re-run. The literature in the introduction was found by searches run on 9 November 2026 and read at the level of abstracts, publisher pages and repository pages; the full texts of the cited papers were not read for this paper.

11 Corrections

- 1 November 2026: the unseen-text control was added after the first results were seen and is labelled post hoc wherever it appears.
- 9 November 2026: first published version.

12 Reproduction

- Data: the English declaration from the NLTK udhr2 collection; Alice from the NLTK Gutenberg selection, characters 5,000 to 35,000; any English text you can show was written after the models were released, in place of our unreleased log.
- Download SmolLM2-360M and Qwen3-0.6B from the Hugging Face Hub at the revisions given in the data section.
- For each text, compute eq. 1 with windows of 512 tokens advancing by 256.
- Confirm with an arithmetic coder that a short passage decodes back to the same tokens.
- Compare with xz at its strongest preset and with PPM-C of order 5 with exclusion, cold and trained on 5 million characters of other Gutenberg books.
- Preregistration SHA-256: 568a182b71d3b95a6414c3ff7f72af999cc0f483658f17580acebc9aac738e90. Plan SHA-256: e37b7bbefb8d5b9b98055a5352ed5fa3f467a29a157dda5129ad02a80dd9434d. Scored run

20261101T091251Z.

- Cost: the model measurements on the declaration, Alice and the answers took 426 seconds on one GPU; the unseen-text run was not timed. The PPM and xz rows run on one desktop CPU.

13 Appendix A. The preregistered forecasts

Each verdict is the pass or fail of a rule written before the run, read off by code. A point forecast holds within 25 percent of its value. Values are read from the study's score file. The unseen-text control has no forecast and is not listed.

Table 2. Preregistered forecasts for the compression part of the study and their outcomes.

Forecast	Rule	Measured	Holds
xz on the declaration	3.0	2.83	yes
PPM cold on the declaration	2.4	2.57	yes
PPM trained on the declaration	2.1	1.99	yes
SmolLM2 on the declaration	1.2	0.27	no
Qwen3 on the declaration	1.0	0.57	no
xz on Alice	2.9	3.12	yes
PPM trained on Alice	2.1	2.14	yes
SmolLM2 on Alice	2.0	0.89	no
SmolLM2 above cold PPM, of 9 other translations	at least 3	0	no
Qwen3 below cold PPM, of 9 other translations	at least 6	8	yes
SmolLM2 payback against xz, characters	at least 1 billion	2.6 billion	yes
Trained PPM payback against xz, characters	at most 200 million	11.9 million	yes
Arithmetic round trip exact	yes, both models	yes, both	yes
Capital city, bits, Qwen3	at most 4	0.95	yes
One-digit sum, bits, Qwen3	at most 4	1.29	yes
Three-digit sum, bits, Qwen3	at least 10	9.11	no
Wrong minus right answer, bits, Qwen3	at least 10	8.08	no
PPM trained on Gutenberg, cheapest task mean, bits	at least 20	20.4	yes
PPM with a stored answer table, stored answers, bits	at most 2	11.2	no
PPM with a stored answer table, unseen answers, bits	at least 15	4.9	no
Share of trained PPM gain removed by shuffled characters	at least 0.8	2.74	yes

The answer forecasts are scored on Qwen3 because the scoring code picks the model with the lower cost on capital cities. The two rows on other translations compare each model with cold PPM on nine non-English translations of the declaration, which carry the same contamination as the English rows. The stored-table test did not separate lookup from computation as forecast. A gain share above 1 means the shuffled training made PPM worse than starting cold.

14 Glossary

Each term is defined where it is first used in the text. The same definitions, with the field each comes from and what each excludes, are collected here.

- **arithmetic coding**. A method that turns a model's probability for each next symbol into a single binary number, so a symbol given probability p costs close to minus log2 p bits, and a decoder with the same model recovers the text exactly. Field: data compression. Excludes codes that give each symbol a fixed

whole number of bits, such as Huffman codes.

- **memorisation**. A model's ability to predict a specific text because that text, or close copies of it, was in its training data, as distinct from predicting text of the same kind it has not seen. Field: machine learning; in evaluation also called training-data contamination. Excludes any claim about how the model stores the text internally.
- **bits per character**. The length of a compressed text in bits divided by the number of characters in the original; lower is better, and 8 is the cost of plain one-byte text. Field: information theory and data compression. Excludes bits per token or per byte, which divide by a different count.
- **preregistration**. A dated written plan for a study, fixed and hash-locked before any result is seen: the question, the data, the statistic, the controls, the forecasts and the kill conditions. Field: research method (metascience). Excludes a plan revised after looking at the outcome, and a protocol written only for the record after the run.
- **post hoc control**. A control added after the first results were seen; it can test an explanation but carries less weight than a control fixed in advance, because the choice of control could follow the result. Field: research method. Excludes controls named in the preregistration.
- **token**. The unit a language model reads and predicts, a whole word, part of a word or a single character taken from a fixed vocabulary of tens of thousands of pieces. Field: natural language processing. Excludes a character or a byte.
- **ideal code length**. The sum over a text of minus log2 of the probability the model gave to each symbol that came next; a real arithmetic coder comes within a few bits of it. Field: information theory. Excludes a measured file size, and any cost of storing the model.
- **xz**. A general-purpose file compressor using the LZMA method, which replaces repeated strings with references to earlier copies; here at its strongest preset. Field: data compression software. Excludes any model of language beyond repeated byte strings.
- **PPM**. Prediction by partial matching, a compressor that predicts each character from counts of what followed the same preceding characters, falling back to shorter contexts when a longer one has not been seen; here method C with exclusion and contexts of up to 5 characters. Field: data compression. Excludes models with learned weights, such as neural language models.
- **payback length**. The number of characters a model must compress before the bits it saves over another compressor add up to the size of the model itself, when the model has to be sent along with the message. Field: our usage, after the two-part code of minimum description length. Excludes a compression ratio on one text, which ignores the cost of the model.
- **control**. A version of the data with the structure under test removed (for example shuffled or pooled), scored the same way as the real data, so the real score has something to beat. Field: statistics and experimental design. Excludes a comparison with a different dataset or with a published number.
- **verdict**. The pass or fail of one preregistered rule set against the run's numbers, read off by code. Field: research method (our usage). Excludes an interpretation written after reading the numbers.

15 References

[1] Deletang, G., Ruoss, A., Duquenne, P.-A., Catt, E., Genewein, T., Mattern, C., Grau-Moya, J., Wenliang, L. K., Aitchison, M., Orseau, L., Hutter, M. and Veness, J. (2024). Language modeling is compression. International Conference on Learning Representations (ICLR 2024). arXiv:2309.10668

[2] Huang, Y., Zhang, J., Shan, Z. and He, J. (2024). Compression represents intelligence linearly. arXiv:2404.09937

[3] Li, Y., Guo, Y., Guerin, F. and Lin, C. (2024). Evaluating large language models for generalization and robustness via data compression. arXiv:2402.00861

[4] Ben Allal, L., Lozhkov, A., Bakouch, E. and others (2025). SmolLM2: when smol goes big, data-centric training of a small language model. arXiv:2502.02737. Model: https://huggingface.co/HuggingFaceTB/SmolLM2-360M, Apache-2.0.

[5] Qwen Team (2025). Qwen3 technical report. arXiv:2505.09388. Model: https://huggingface.co/Qwen/Qwen3-0.6B, Apache-2.0.

[6] Cleary, J. G. and Witten, I. H. (1984). Data compression using adaptive coding and partial string matching. IEEE Transactions on Communications 32(4), 396-402. doi:10.1109/TCOM.1984.1096090

[7] United Nations (1948). Universal Declaration of Human Rights, English text and translations as distributed in the NLTK udhr2 corpus. Source address of the download not recorded; licence not established.

[8] Project Gutenberg texts as distributed in the NLTK Gutenberg selection. https://www.gutenberg.org. Public domain in the United States.