Institute of Information Security

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Question and Answer Dataset Corpus for Cryptography

Tilman Dietz

Course of Study: B.Sc Data Science

Examiner: Prof. Dr. Ralf Küsters

Supervisor: Mayar Elfares M.Sc.

Commenced: January 22nd, 2025

Completed: July 28th, 2025

Abstract

Large language models (LLMs) have demonstrated remarkable performance across a wide range of Natural Language Processing tasks, like text summarization, machine translation, and question answering. Their architecture, which is grounded in large-scale transformer models that are trained on diverse textual corpora, enables LLMs modeling complex linguistic structures and contextual relationships. These capabilities have driven innovation in computer science-related tasks like code generation, code repair, and code translation as well as in language-centric applications. Despite their general proficiency, LLMs continue to struggle in domains like mathematics and cryptography that require structured multi-step reasoning. In the field of cybersecurity, LLMs are used for threat detection, incident response, and the training of employees and end users. However, their ability to perform cryptographic tasks remains under-explored.

To address this gap, we introduce CryptoQA, a high-quality, large-scale dataset containing over one million question-answer-pairs covering all major areas of cryptography and related topics. The data set is curated from large-scale web-scraped cryptographic contents such as textbooks, research articles, and educational materials. Each entry is enriched with metadata, providing scalable and consistent annotations without the cost of human labeling. In addition to curating the dataset, we evaluate state of the art LLMs on cryptographic knowledge and their reasoning abilities. Our initial evaluations of state of the art models reveal significant performance gaps, highlighting the need for continued research and specialized training to enable effective LLM integration in cryptographic applications.

Kurzfassung

Large Language Models (LLMs) haben eine bemerkenswerte Leistungen in einer Vielzahl von Aufgaben der natürlichen Sprachverarbeitung, wie Textzusammenfassung, maschinelle Übersetzung und Beantwortung von Fragen,gezeigt. Ihre Architektur, die auf groß angelegten Transformationsmodellen basiert, welche auf verschiedenen Textkorpora trainiert werden, ermöglicht LLMs die Modellierung komplexer linguistischer Strukturen und kontextueller Beziehungen. Diese Fähigkeiten haben Innovationen unter anderem in der Informatik, wie z. B. die Generierung von Code, die Reparatur von Code und die Übersetzung von Code, sowie in sprachzentrierten Anwendungen vorangetrieben. Trotz ihrer allgemeinen Fähigkeiten haben LLMs in Bereichen wie Mathematik und Kryptographie , die strukturiertes, mehrstufiges Denken erfordern, weiterhin Schwierigkeiten. Im Bereich der Cybersicherheit werden LLMs für die Erkennung von Bedrohungen , die Reaktion auf Zwischenfälle und die Schulung von Mitarbeitern und Endbenutzern eingesetzt. Ihre Fähigkeit, kryptografische Aufgaben auszuführen, ist jedoch noch wenig erforscht.

Um diese Lücke zu schließen, stellen wir CryptoQA vor, einen qualitativ hochwertigen, groß angelegten Datensatz mit über einer Million Frage-Antwort-Paaren, die alle wichtigen Bereiche der Kryptographie und verwandter Themen abdecken. Der Datensatz wurde aus umfangreichen, im Internet gescrapten kryptografischen Inhalten wie Lehrbüchern, Forschungsartikeln und Lehrmaterialien zusammengestellt. Jeder Eintrag ist mit Metadaten angereichert, so dass skalierbare und konsistente Annotationen ohne die Kosten der menschlichen Beschriftung möglich sind. Zusätzlich zur Kuratierung des Datensatzes evaluieren wir moderne LLMs in Bezug auf kryptografisches Wissen und ihre Schlussfolgerungsfähigkeiten. Unsere ersten Auswertungen der dieser Modelle zeigen erhebliche Leistungslücken auf, was den Bedarf an fortgesetzter Forschung und speziellem Training für eine effektive Integration von LLMs in kryptografische Anwendungen verdeutlicht.

Contents

List of Figures

List of Tables

Listings

List of Abbreviations

AI artificial intelligence. 23, 33

COT chain of thought. 26

FFNN feed-forward neural network. 22

GLUE General Language Understanding Evaluation. 34

LLM large language model. 3, 17, 19, 31, 38, 43, 61, 65

LM language model. 19

MoE Mixture of Experts. 22, 38

NLP Natural Language Processing. 17, 19, 33

NN neural network. 19

PTS pre-training from scratch. 25

QA question answer. 18, 31, 35, 43, 65

RAG Retrieval-Augmented Generation. 26, 38

RNN recurrent neural network. 19

ROUGE Recall-Oriented Understudy for Gisting Evaluation. 28

SOTA state of the art. 17, 31

1 Introduction

Large language models (LLMs) have demonstrated extraordinary capabilities across a wide spectrum of Natural Language Processings (NLPs) tasks, including text summarization, machine translation, sentiment analysis, and question answering. LLMs, which are typically based on the transformer architecture and trained on massive corpora of diverse text from different domains, can model complex linguistic patterns and contextual dependencies[21].

In the domain of language-centric applications, LLMs have facilitated significant advancements such as real-time machine translation, contextual auto-completion, intelligent email drafting, grammar correction, and generating dialog. Their utility in interactive systems such as virtual assistants, chatbots, and adaptive learning platforms continues to grow, which is mainly driven by their capacity to understand nuance, semantics, and intent in human language.

Similar to language-centric applications, LLMs have shown proficiency in a range of code-related tasks, such as code auto-completion, translation between programming languages, bug detection and code repair, as well as the generation of executable code directly from natural language descriptions[20]. These capabilities are enabling more intuitive and accessible software development workflows, empowering both experienced developers and non-programmers to interact with code through natural language interfaces.

LLMs exhibit promising, albeit still rapidly evolving, capabilities in performing compositional and commonsense reasoning tasks, which have been traditionally considered to be challenging for statistical language models[8, 51]. These developments are fostering a surge in downstream applications that leverage LLMs for real-world applications in different domains such as in cybersecurity.

1.1 Problem Statement

In cybersecurity, LLMs have gained more and more use-cases such as threat detection, incident response, cyber forensics, FAQ automation, security training and awareness [14]. However, not all fields of cybersecurity are currently benefiting from LLMs. Recent research has shown that even state of the art (SOTA) model like ChatGPT-4o, GPT-o1 and DeepSeek-R1 struggle with simple cryptographic tasks [26]. These problems are caused by a lack of specialized training data as well as by the nature of arithmetic problems, which require multi-step reasoning to arrive at the correct solution[51]. Furthermore, cryptography is an inherently difficult task requiring solid foundations in number theory, algorithms and complexity theory, probability, algebra as well as logical reasoning[25]. Therefore, extensive dataset benchmarking is needed before LLMs can be used in assisting researchers in developing and implementing cryptographic applications.

1.2 Contribution

To solve this problem, we address the lack of specialized training data by curating a high-quality dataset with question answer (QA)-pairs from all areas of cryptography and related topics. For each QA-pair, we add some metadata. Instead of using expensive human-generated metadata, we automate the generation by leveraging the LLM DeepSeekV3. By doing this, we can generate more QA-pairs compared to conventional approaches.

More specifically, we provide `CryptoQAa`, a high-quality Question-Answering dataset with over 1 million questions in all areas of cryptography and related topics. We curated this dataset by leveraging DeepSeekV3 to generate QA pairs based on a large-scale web scraping of cryptographic content, e.g. textbooks and proceedings. We further provide a benchmark to evaluate LLM on their knowledge of cryptography and test it on SOTA LLMs.

1.3 Structure

The remaining work is structured as follows.

In Chapter 2, we look deeper into the foundations of LLMs. In Chapter 3, we provide an overview of existing QA-datasets and benchmarks for LLMs. In Chapter 4, we explain our curation process and provide an overview of CryptoQA. We furthermore provide a benchmark to evaluate LLMs on their cryptographic knowledge. We benchmark DeepSeekV3 and DeepSeekR1 on CryptoQA in Chapter 5. Chapter 6 details the results and limitations of this study, as well as suggestions for future research. Chapter 7 concludes the thesis.

2 Foundations

In this chapter, we go into the foundations of this work. First, we explain the architecture of large language models (LLMs) in Section 2.1. After that, we delve into the training of LLMs in Section 2.2. Moreover, we go into utilization and limitations of LLMs in Section 2.3. Finally, we provide an overview over evaluation of LLMs

2.1 Large Language Models

A (pretrained) language model predicts, given a sequence of words, the likelihood of following words based on its beforehand learned parameters[22]. LLMs differ from language models (LMs) primarily in the number of parameters, albeit there is no formal boundary distinguishing the two. Nevertheless, LLMs have at least hundreds of millions of parameters[22]. Modern LLMs like GPT-3[7] and DeepSeekV3[9] have more than 100 billion parameters.

To be able to process text, the input text has to be split into tokens[22]. Tokens are the smallest units of text that a model processes [34]. Tokenization can happen on a character, sub-word, word level or a mix between them[34]. These tokens are then embedded into feature vectors $x_1, ..., x_n$ from a given d-dimensional feature space[22].

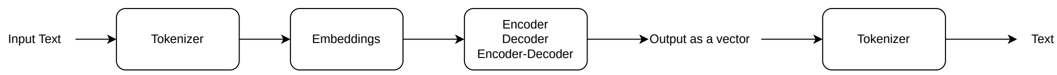

Figure 2.1: From input to output

In the encoder-decoder section the embeddings are then processed by a neural network (NN). The NN returns a vector of probabilities for certain tokens. Based on these probabilities, the final text is sampled and post-processed in to a string [45].

2.1.1 Encoder-Decoder architecture

The encoder-decoder architecture laid the foundation for modern LLMs and has been used for many sequence-to-sequence Natural Language Processing (NLP) tasks like abstractive summarization, machine translation and abstractive question answering[22].

Recurrent neural network (RNN) based language models read the input sequentially. For each token of the input sequence $\mathtt{x_i}$, the hidden state of the encoder is updated using the previous state h_{i-1} and the input token x_i

$$h_i = f(h_{i-1}, x_i) \tag{2.1}$$

The output of the encoder is called context vector or context variable `c` and is is calculated by

$$c = m(h_1, ..., h_n) \tag{2.2}$$

where m is a mapping function.

The decoder outputs a sequence $y = (y_1, .., y_o)$, based on the context vector c, the previous state of the decoder, and the previous returned tokens. The state of a decoder at step is given by

$$s_{t'} = g(s_{i-1}, y_{i'-1}, c) \tag{2.3}$$

The hidden state is then used by an output layer, where the conditional distribution of a token is given by

$$P(y_{t'}|y_{t'-1}, .., y_1, c) = softmax(s_{t-1}, y_{t'-1}, c) \tag{2.4}$$

While the encoder-decoder architecture was an important step in NLP, condensing the information from the whole input sentences into a singular context vector leads to information loss, especially for large inputs. Furthermore, encapsulating natural language's long-range dependencies between tokens represents a major challenge. In addition to this, due to the sequential nature of the encoder-decoder architecture, parallelization of training and inference is very difficult[22].

2.1.2 Attention

To reduce the bottleneck of encoding a whole sentence into one single vector, Bahdanau et al introduced attention mechanisms[5]. Attention mechanisms enable LMs to model long-range dependencies in the input and output of the model[46].

Compared to the default RNN-encoder-decoder approach, the hidden state of the encoder at time step `t` is no longer depending on a constant context vector, but on a context vector for each step.

$$s_i = g(s_{t-1}, y_{t'-1}, c_t) \tag{2.5}$$

with the conditional probability of a token.

$$P(y_{t'}|y_{t'-1}, .., y_1, x) = softmax(y_{t-1}, s_t, c_t) \tag{2.6}$$

The context vector c_t depends on the hidden states computed by the encoder $(h_1, ..., h_n)$.

$$c_t = \sum_{k=1}^{n} w_{tk} h_k \tag{2.7}$$

The weights w_{tk} are measuring how well the input around position k and the output at position match[5]. The weights depend on the previous hidden state s_{t-1} and the hidden state of the encoder after reading the k token h_k.

To extend this model introduced by Bahdanau et al.[5] the attention model is often represented as memory bank with key-value pairs and a query[34]. Each input vector in split into a key-vector k_i and value vector v_i. The query, previously the previous hidden state, is know represented as q_i.

For each key k_i a score function α returns

$$a_i = \alpha(q, k_i) \tag{2.8}$$

The score is used to calculate the weights using the softmax-function[22, 34].

$$w_i = \frac{\exp(a_i)}{\sum_{j=1}^{n} \exp(a_j)} \tag{2.9}$$

The output of the attention layer is

$$o = \sum_{k=1}^{T} w_k v_k \tag{2.10}$$

2.1.3 Self-Attention

As an extension of attention mechanisms, self-attention mechanisms project each input vector into a query-vector $q_i = x_i W_q$, a key-vector $k_i = x_i W_k$ and a value-vector $v_i = x_i W_v$. W_q, W_k and W_v are learnable weight matrices[22]. The final attention matrix is given by

$$attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.11}$$

2.1.4 Transformer architecture

While Attention mechanisms resolved the issue of compressing all the input into a single vector, LMs still suffered from being unable to parallelize the language model due to the sequential nature of the RNN-encoder-decoder architecture[1]. To combat these limitations, Vaswani et al.[46] introduced the transformer architecture.

In the transformer architecture, the sequential processing due to the RNN-based processing was eliminated in favor of solely relying on attention mechanisms to model dependencies and context[46]. Instead of reading the input one by one, the processing of the input could now be parallelized due to the transformer architecture. This allowed for faster training of models compared to RNN-based language models. The transformer architecture laid the foundation for many pre-trained LLMs like BERT[10],T5[37], the GPT[7] family and DeepSeekV3[9].

The original transformer-model by Vaswani et al. used a encoder-decoder architecture[46]. The encoder as well as the decoder stacks consisted out of several transformer layers[46].

Transformer Layer

A transformer layer(Figure 2.2) receives the input as d x n matrix of the embedded tokens combined with positional encodings. The layer consists of two sublayers. In the first sub-layer, a multi-head self-attention layer is fed the input encodings with the positional encodings. The second sub-layer is a feed-forward neural network. After both sub-layers is a layer-normalization-layer to normalize the outputs. The layer-normalization-layers receive the output of the sub-layers as well as the input of the sub-layers[46].

The encoder in the original Transformer by Vaswani et al., as described above, is usually referred to as the transformer block. However, the decoder slightly deviated from this general framework by incorporating the output of the encoder into every layer[46]. Other LLMs may use other attention functions, normalization layers, or a different structured feed-forward neural networks (FFNNs).

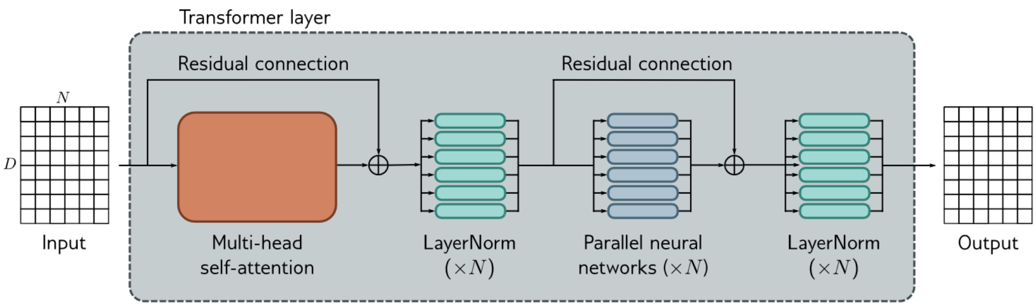

Figure 2.2: The transformer block by Vaswani et al.[46],picture by [36]

Modifications to the Transformer architecture

While Vaswani et al. used an encoder-decoder architecture[46], many other pre-trained LLMs use either only a decoder, like the GPT-family[7]and the DeepSeek-family[9], or only an encoder, like BERT[10].

Mixture-of-Experts

In an Mixture of Experts (MoE) transformer, the FFNN is replaced by an MoE-layer[42]. The MoE-layer consists of many FFNNs expert models and a gating neural network. The gating network calculates an expert rating for the specific input and selects the k-best experts.

2.1.5 Decoding of tokens

After processing a whole token, the transformer returns a probability distribution of tokens[36]. The selection strategy directly affects the quality of the final output text. The sampling of the output tokens can be done in a greedy or randomized way.

The simplest greedy strategy is simply selecting the most probable token. However, this strategy may result in outputs with a lower probability compared to the optimal solution due to some local maxima. Furthermore, texts sampled with a greedy strategy are often considered awkward and repetitive[19]. To alleviate this issue, beam search can be used, which keeps the n-most frequent solutions after each prediction[55].

While greedy search strategies can produce fitting outputs in tasks where the input is heavily dependent on the input, like translation and summarization, they produce repetitive text[19, 55]. Sampling-based methods combat this issue by sampling the tokens from the whole vocabulary. The most prominent sampling methods are Temperature sampling, Top-k sampling and Top-P sampling. In temperature sampling, the probability of selecting a word depends on the temperature. If the temperature is close to 0, it is behaving like greedy search. Top-k extends this by only considering the tokens with the k highest probabilities. As an improvement for Top-k sampling, Top-p sampling considers only tokens until the cumulative probability p is reached[19].

2.2 Training of Large Language models

The training of LLMs is divided into a pre-training phase and a fine-tuning phase. In the pre-training phase, the LLM is trained on large volumes of unlabeled data using self-supervised learning, while in the fine-tuning phase, it is trained supervised on a small, labeled dataset.

2.2.1 Pre-training

In the pre-training phase, the model is trained on an enormous amount of unlabeled text data using self-supervised learning[21]. Pre-training consists of preparing the training corpus, generating the vocabulary, designing the pre-training tasks, choosing the pre-training method and choosing pre-training dynamics. Self-supervised learning is a pivotal advancement in artificial intelligence (AI). Previous to the application of self-supervised learning, LM were trained on labeled datasets, hence the name supervised learning. However, obtaining enough human-annotated instances to achieve a good performance is expensive and time-consuming. This is especially problematic in domains where the data is sparse, like legal or medical applications. As LMs can only learn what they have in their training data, they often generalize poorly and suffer from false correlations[21]. The goal of self-supervised learning is to learn meaningful representations and improve the generalization abilities of the model[22]. The loss for self-supervised learning is the weighted sum of all pre-training tasks[22].

$$\mathcal{L}_{SSL} = \sum_{i \in \text{Tasks}} \lambda_i \cdot \mathcal{L}_{\text{Task}_i} \tag{2.12}$$

Pre-training corpora

LLMs are trained on vast amounts of unlabeled text data. Increasing the pre-training data set has been shown to improve the performance of the LLM[4]. However, not only does the size influence performance, but also the type of data. Data sources may be scraped web pages from datasets like Common Crawl, C4[37], or the whole Wikipedia but also books as well as code from GitHub and StackExchange[22].

While there are more than 300 billion web pages in Common Crawl, most of them are not in natural language. Filtered datasets like C4[37] may mitigate some of the drawbacks of scraping web pages by removing HTML code, offensive language, duplicates of text and sentences, which do not end on a terminal punctuation mark. However, the quality of the model still depends on the quality of the content of the scraped web pages. Especially early LLMs included Wikipedia in their pre-training dataset as it is consistently formatted and usually well edited. In addition to this, Wikipedia also contains data for many languages[22].

While Wikipedia provides an excellent starting point for formal language, it is primarily written in the third person using formal language and therefore does not usually capture how humans think and interact with other humans or the environment. In general book are also excelling at modeling long-term dependencies[22].

As the general capabilities of generative models have increased, the generation of source code has become increasingly popular. Interestingly, training on source code from GitHub and StackExchange not only improved its performance in generating code but also improved the performance on other logical reasoning tasks[22].

Generating the vocabulary

Before the actual training of LLMs can begin, a vocabulary needs to be generated first. For transformer-based LLMs, character or sub-words embeddings are chosen over word embeddings as they offer a lower vocabulary size, which also lowers the model size. Furthermore, sub-word embeddings can represent even words that are not initially in the vocabulary and can encode fine-grained information at a sub-word level into the representation of words[21]. Usually, for tokenizers, the size of the vocabulary has to be chosen before training. If the vocabulary size is too small, more words will be split into many sub-words, inhibiting model learning and increasing the pre-training time[21]. A too large vocabulary results in more words being represented as one token. However, this also increases the model size.

Pre-training tasks

During pre-training, a LLM has to solve one or more pre-training tasks. In Casual Language Modeling, a model predicts the next word based on the preceding or succeeding words[21]. As an improvement of this, in Masked Language Modeling some input tokens are masked. The models have to predict those masked tokens[10].

Pre-training methods

There are many ways to design the training of LLMs. The training method describes how the model is pretrained. The choice of the training method significantly influences the cost and duration of the pre-training phase. It has an influence on the overall performance of the LLM and should be fitting for the task[22].

In pre-training from scratch (PTS) models are trained on enormous volumes of unlabeled text data. Usually, all parameters of all layers, including the embedding layer, are initialized randomly and then learned by minimizing the loss of the pre-training task[21, 22]. While this approach does not rely on prior knowledge, it has an enormous computational demand.

In Continual Pre-training an existing pre-trained LM is used as a base to further pre-train the model using a domain specific dataset. It is often used to create domain specific language models. While Continual Pre-training is cheaper than PTS, it suffers from the lack of a domain-specific vocabulary [21].

In Simultaneous Pre-training a model is pre-trained from scratch using a combination of a general text dataset as well as a domain specific dataset[21].

Rather than pre-training a LM for general or domain-specific applications, in Task Adaptive Pretraining, a language model is further pre-trained on additional task-specific unlabeled data[16]. Task Adaptive Pretraining is usually done after PTS or Continual Pretraining[21].

The previous methods utilized only self-supervised learning. In contrast, Knowledge inherited Pre-training integrates Knowledge distillation from existing LMs, resulting in faster convergence compared to approaches that solely rely on self-supervised techniques[21].

Choosing the pre-training dynamics

Selecting the parameters for training influences the duration of pre-training as well as the performance of the model. Using larger batch sizes, performing more pre-training steps and using longer input sequences enhances the performance of the LLM[30]. To faster converge, it has been recommended to use different learning rates in different layers, as well as to increase the learning rate in the early phases of pre-training[21, 52].

2.2.2 Fine-tuning

While in the pre-training phases a LLM gained general knowledge about language, its performance on downstream tasks requiring task-specific knowledge is still suboptimal. To improve the performance of LLMs after pre-training, fine-tuning of the parameters of the LLM is required. During the fine-tuning stage, the models weights are adapted to improve the performance on specific downstream tasks. In contrast to the pre-training stage, only labeled data is used to adapt the LLM to the specific downstream application[22]

2.3 Utilization and Limitations

In this section, we look into certain techniques, settings, and design choices to improve the performance of models without expensive pre-training or fine-tuning. We furthermore provide an overview over challenges and limitations of LLMs.

2.3.1 Utilization

Prompt engineering

Prompt engineering refers to the careful creation of prompt templates to optimize performance on a downstream task[22]. For successful prompt engineering we need to consider the used model, the shape of the prompt, as well as the engineering of prompt templates and answers. The generation of prompt templates can be done manually or automatically.

The prompt are usually prefix prompts and cloze prompts. In prefix prompts, a slot for the answer is placed behind the input as well as the prompt referencing it. Prefix prompts have shown good performance in tasks establishing a simple, apparent connection between the response and the input, like machine translation. In a cloze prompt a slot for the answer is encapsulated by the prompt itself[22].

Prompt engineering requires the precise formulation of the downstream task, potential background information, and the answer format[22].

Prompt setting

As described in the preceding section, careful prompt engineering can significantly enhance the performance of LLMs on downstream tasks. Additionally, providing examples within prompts has been demonstrated to further improve results.

In a zero-shot setting, the prompt contains only the task description. A prompt containing one or a few examples is called a one-shot or few-shot prompt. LLMs have demonstrated that supplying a model with examples of the given task can improve their performance. The ability to learn from provided examples is also referred to as In-context learning[55].

Chain of thought

To improve the reasoning abilities of LLMs, chain of thought (COT) approaches have been proposed by Wei et al[51]. COT is a prompting and reasoning technique where the prompted model generates intermediate logical steps leading to a conclusion. It enhances accuracy for complex tasks, particularly in math, logic, multi-hop QA, and code generation. While COT-approaches initially extended In-context learning approaches by providing a rationale in the example prompts[51], further research suggests that such COT approaches can be elicited from prompts which do not provide examples but only the request to think step by step [23]. However, recent evaluations suggested that COT only benefits models that have an enormous size[51].

Retrieval-Augmented Generation

To enhance factual correctness of LLMs, Retrieval-Augmented Generation (RAG) has been proposed. Instead of relying solely on pre-trained internal parameters like conventional approaches, RAG incorporates an external knowledge retrieval mechanism to access relevant documents or passages at inference time[53]. These retrieved documents are concatenated with the input query and fed into a generative model, which conditions its response on both the original prompt and the retrieved evidence[22]. This method has demonstrated superior performance compared to conventional approaches in tasks such as open-domain question answering and domain-specific chat systems, particularly in scenarios where up-to-date or long-tail knowledge is required[12]. By decoupling static knowledge from model weights, RAG enables scalable and dynamic updating of information sources without retraining the underlying model, thereby offering a more efficient and interpretable alternative to traditional closed-book models [12, 22].

2.3.2 Limitations of LLMs

While LLMs have shown impressive performance in many NLP task, they still exhibit several critical limitations that restrict their applicability in critical domains like cryptography.

Arithmetic computations and logical reasoning

Reasoning and performing arithmetic operations have been challenging for LLMs[15]. This is partly due to the lack of verification in previous work[8]. While approaches like COT have improved performance on reasoning tasks, it remains unclear if this performance is just pattern detection and heuristics[51].

Empirical evaluations have shown that LLMs can predict the first digit of a number but struggle to predict the last digit of the result[15]. Further research suggests that the representation of numbers in LLMs is a limiting factor in arithmetic computations. While LLMs operating with low parameter precision cannot reliably perform arithmetic operations, LLMs utilizing standard precision can perform arithmetic operations for numbers of up to a certain length[13].

Hallucinations

Hallucinations occur if a model produces factually incorrect content, which contradicts the original source material [22]. Hallucination can be broadly classified into input-conflicting, context-conflicting, and factual-conflicting hallucinations[22]. This can have multiple reasons, like knowledge deficiencies, biases in the training data, inaccuracies in the training dataset, overconfident responses, incorrect alignment processes, and sampling strategies.

2.4 Evaluation of Large Language Models

LLMs can be evaluated in an intrinsic or an extrinsic way. Intrinsic Evaluation tests LLMs on its knowledge learned during pre-training. In extrinsic evaluation, LLMs are evaluated on real-world tasks[21].

To perform extrinsic evaluation, benchmarks are used often as they provide sets of datasets, a leader board, as well as metrics for comparison[21]. The metric needs to be chosen in the context of the benchmarked task. In the following, we provide an overview of the metrics that we use in this work.

BLEU

Bilingual Evaluation Understudy (BLEU) is an n-gram based metric initially meant for machine translation systems[35]. It is based on the n-gram precision. A n-gram is a sequence of n words. Matching n-grams between the candidate sentence and the reference sentence are counted and then divided by the total number of n-grams. However, the number of matching n-grams is capped by the number of occurrences in the reference answer. The modified n-gram precision p_n is given by

$$p_n = \frac{\text{number of clipped matching n-grams}}{\text{number of n-grams in the generated text}} \tag{2.13}$$

Between the modified n-gram precisions, the geometric mean is calculated; however, the geometric mean can be modified by giving different weights to the different scores.

Due to the nature of clipping by the modified n-gram precision, answers that are longer than the reference answer are already penalized. To reduce the effect of too short answers, a brevity penalty is introduced[35].

$$BP = \begin{cases} 1 & c > r \\ \exp(1 - \frac{r}{c}) & c \leq r \end{cases} \tag{2.14}$$

The final metric is given by

$$BLEU = BP \cdot \exp\left(\sum_{i=1}^{N} w_i \log(p_i)\right) \tag{2.15}$$

While it is an old, established metric for machine translation, it lacks human-like performance for evaluating generated text and free-text answers. Furthermore, the score is zero if there is no n-gram overlap between the reference sentence and the candidate sentence. Additionally, it is only meant to match human judgment when averaged over the whole corpus.

The parameters are usually chosen to be N=4 and $w_i = \frac{1}{N}$ [22].

ROUGE

Recall-Oriented Understudy for Gisting Evaluation (ROUGE) is a measure initially meant to evaluate text summaries. ROUGE is a package that includes several different methods to automatically evaluate summaries.

ROUGE-N is the n-gram recall between the candidate summary and a set of reference summaries[27].

$$ROUGE - N = \frac{\text{number of all n-grams in all reference sentences and the candidate sentence}}{\text{number of all n-grams in all reference sentences}} \tag{2.16}$$

ROUGE-L is a longest common subsequence based F-measure[27]. Let X be a reference summary of length m and Y a candidate sentence of length n. Let LCS(X,Y) the length of the longest common subsequence of X and Y. The ROUGE-L score is given by

$$R_{LCS} = \frac{LCS(X,Y)}{m} \qquad P_{LCS} = \frac{LCS(X,Y)}{n} \tag{2.17}$$

$$ROUGE-L = \frac{(1+\beta^2)R_{LCS} \cdot P_{LCS}}{R_{LCS} + P_{LCS}} \tag{2.18}$$

BERTscore

BERTscore is a metric designed to evaluate generated text $\hat{x}$ against a reference answer x[54]. It utilizes contextualized embeddings to generate precision P_{BERT}, recall R_{BERT} and F1 score $F1_{BERT}$. The score is calculated with the sum of the cosine similarities between the embeddings of the tokens. The scores are calculated as follows:

$$P_{BERT} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} x_i^T \hat{x_j} \tag{2.19}$$

$$R_{BERT} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} x_i^T \hat{x_j} \tag{2.20}$$

$$F1_{BERT} = \frac{2 \cdot P_{BERT} \cdot R_{BERT}}{P_{BERT} + R_{BERT}} \tag{2.21}$$

The scores are normalized to be in the range of [0,1][54].

BLEURT

Bilingual Evaluation Understudy with Representations from Transformers (BLEURT) is a learned metric for evaluating text generation[41]. Based on a candidate sentence, it predicts a rating of how close the generated answer is to the reference answer. This is done by first computing the contextualized embeddings of the word embeddings of the answers using BERT or another BERT-based model, and then using a linear layer to predict the final scores.

3 Related Work

In the following chapter, we provide an overview of related work. In Section 3.1 we dive into other existing question answer (QA)-datasets in the domain of mathematics and cybersecurity. For this, we only consider datasets that were primary meant for training or evaluation of one specific task. Multitask benchmarks will be covered in Section 3.2

3.1 Question-answering datasets

In this section, we delve into QA-datasets. As they can be used for training and evaluating models, there has been a need for high-quality QA-datasets. More than 80 QA-datasets have been released in 2021 and 2022[38]. In the following, we go over existing cryptographic QA datasets and benchmarks. After that, we look into mathematical and reasoning datasets. Finally, we provide an overview of other QA-datasets within the cybersecurity domain.

3.1.1 Existing Cryptographic Benchmarks

The `CipherDataset` by Noever is the first published dataset we could find. It contains 654 from 51 unique text and 13 algorithms. Noever was the first to evaluate the decryption capabilities. He found that after a few iterations of prompting GPT-4 was able to decrypt simple ciphers [33].

The first available dataset covering theoretical questions about cryptography as well as the application of encryption schemes was published by Barman[6]. As the primary focus of Barman was not the QA dataset but fine-tune large language models (LLMs) to solve cryptographic problems, it includes only 510 training and 50 testing QA-pairs. Barman used a similar approach to curate QA-pairs by leveraging GPT-3.5 to generate the questions, answers and other labels based on textbooks about cryptography. It contains questions about cryptography as well as questions, where the model should encrypt, decrypt or perform other calculations which serve as a foundation for cryptographic operations[6].

Another QA dataset is CipherBank by Li et al[26]. It contains 2358 problems, which were encrypted using nine different symmetric encryption algorithms. To evaluate the result of LLM, they proposed to use a Levenshtein-distance based score.

Recently Maskey et al.[32] published a benchmark which evaluates LLMs on their ability to perform cryptanalysis. For that, they generated several plaintexts and encrypted them with 9 different encryption algorithms, including classical encryption algorithms like the Caesar cipher as well as modern encryption algorithms like AES and RSA. However, they only used one key for each encryption algorithm. They used their dataset to evaluate 6 different state of the art (SOTA) large language models. In their evaluation, they found that LLMs could detect and decrypt simple encryption algorithms like Caesar cipher and altbash but struggled with more difficult algorithms like Vigenere, RSA, and AES.

3.1.2 Mathematical QA datasets

Although there have not been many published datasets about cryptography, more QA datasets have been published about mathematics.

One of the earliest released QA datasets released was by Kushman et al. in 2014[24]. The dataset contains 514 algebra words problems. The questions were originally posted by students on the website `algebra.com` to get help solving them. Members of the community could then help them solve the questions. The cleaning of the data was done by volunteers on Amazon Mechanical Turk. The final QA pairs are formatted in a defined template.

In 2015 Roy et al. published the *IL* dataset. The dataset contains 1404 QA-pairs that were collected from `www.k5learning.com` and `www.dadsworksheets.com`. The questions were elementary level math problems where an operation was performed two or three times[40]. In 2016, they published the *Commoncore* dataset[39]. It contains 600 multi-step arithmetic problems, like addition followed by subtraction or addition and multiplication. The problems were retrieved from `www.commoncoresheets.com`

Another QA dataset with a focus on mathematics is AQuA-(RAT) by Ling et. al[28]. It contains over 100 000 samples with multiple choice questions, answers as well as a rationale for the answer. The QA pairs were sourced in two ways. First, they collected 34202 seed problems with a wide variety in topic and difficulty levels. These questions are sourced from exams like the GMAT and the GRE. Secondly, they used crowdsourcing to generate more questions. The Turkers were asked to create a new, similar question based on a provided seed question.

However, this dataset contains quality issues like incorrect solutions, problems, which can only be solved by brute-forcing and rationales that do not contain an adequate amount of steps to solve the problem[3]. To solve these issues Amini et al. introduced MathQa, a dataset containing math word problems[3]. The problems were based on existing problems from AQuA. They classified the documents into different mathematical domains to limit the number of possible operations to solve the problem. At last, crowd workers were asked to generate to annotate those problems with a formal operation program with the solution to the problem. In total, MathQa contains 37k multi-choice math word problems[3].

GSM8K is a dataset containing 8.5k middle-school level math questions[8] split into 7500 training questions and 1000 testing problems. It was sampled by independent contractors, who were asked to generate questions with a corresponding solution. The solutions were written in natural language as they believed that it corresponds better to the internal structure of LLMs. The correctness of the solution was checked by other contractors.

3.1.3 Other cybersecurity datasets

In the domain of cybersecurity there is SecQa[31]. It contains multiple-choice questions generated from a textbook by GPT-4. However, it does not contain questions about cryptography but only about computer security. It does also only contain 242 multiple-choice questions. The samples were refined also using GPT-4. The QA pairs contained a question, four answers the correct answer as well as an explanation.

3.2 Benchmarks

In this section, we provide an overview of existing benchmarks for LLMs.

Big Bench

BIG-bench[43] is a collection of more than 200 natural language tasks. This benchmark covers a wide variety of Natural Language Processing (NLP)-tasks that are believed to be too hard for current LLMs. More than 400 people from over 100 institutions collaborated to create this benchmark. The following tasks are especially relevant for the evaluation of cryptographic capabilities, as they evaluate models on their ability to perform logical reasoning, perform arithmetic computations, as well as perform more complex mathematical computations.

1. abstraction_and_reasoning_corpus This task is based on the Abstract and Reasoning Corpus, a general artificial intelligence benchmark which is targeted at humans as well as artificial intelligences (AIs).
2. arithmetic: This benchmark tests the abilities of models to perform basic arithmetic operations. A result is correct if it matches exactly with the reference answer.
3. chinese_remainder_theorem : This task tries to measure the ability of a model to perform basic number theory problems as well as understand a given task.
4. cs_algorithms: This task tries to evaluate the abilities of models to solve problems that require recursion or dynamic programming to solve.
5. elementary_math_qa This task evaluates the ability of models to interpret and solve mathematical problems written in natural language. In this task, the model has been given a multiple-choice question.
6. identify_math_theorems In this task models are evaluated on their ability to identify purported mathematical theorems at the collegiate level and respond with the correct theorem.
7. mathematical_induction This task evaluates the capabilities of models to understand proofs by induction and detect if something in the proof is missing.
8. matrixshapes In this task the models have to predict the shapes of a matrix after performing certain matrix operations.

9. modified_arithmetic In this task, the model has to perform a modified version of an operation, where the model has to add one after performing the operation. This task is evaluated in a few-shot setting.
10. multistep_arithmetic In this task, a model has to perform multistep arithmetic operations.
11. operators In this task, a model is given a description of an operator in natural language. Then, the model has to apply the operator on an input.
12. word_problems_on_sets_and_graphs In this task, models are tested on how good they can actually understand a situation.

GLUE

General Language Understanding Evaluation (GLUE) is another popular language consisting of nine language understanding tasks. The tasks were created using existing labeled datasets and benchmarks, like the SQuaD and the Semantic Textual Similarity Benchmark [48].

SuperGLUE

As an extension of GLUE, Wang et al. proposed SuperGLUE[47]. Compared to GLUE, it contains more challenging tasks, more diverse task formats, comprehensive human baselines, improved code support as well as more refined usage rules. The tasks were designed to be challenging and too difficult for SOTA models in 2019, but solvable by college-educated English speakers[47].

MATH

MATH by Hendrycks et al.[18] is a challenging benchmark covering 7 subjects and various difficulties. It contains 12,500 challenging problems QA-pairs sourced from high-school math competitions, split into 7500 training problems and 5000 test problems. It covers Prealgebra, Algebra, Number Theory, Counting and Probability, Geometry, Intermediate Algebra, and Precalculus[18]. Each problem is assigned a difficulty from 1 to 5, which follows the AoPS. The questions are formulated using LaTeX-syntax. The solutions have special formatting requirements that allow for the evaluation using an exact match.

MMLU

Massive Multitask Language Understanding (MMLU) is a multi-task benchmark covering 57 distinct subjects in different domains like abstract algebra, nutrition and world religions[17]. The collection of questions was done by students from free online sources.

4 Methodology

In this chapter, we elaborate on our curation process of `CryptoQA`, our novel cryptography dataset. We curated `CryptoQA`, by acquiring high-quality data from academic sources and then using DeepSeekV3 to generate question answer (QA)-pairs.

The chapter is structured as follows. We describe our procedure for acquiring large volumes of data, as well as our choices during the selection of sources in Section 4.1. In Section 4.2, we elaborate on our pre-processing steps, which were necessary to generate the question-answering pairs. In Section 4.3, we explain how we generated the question-answer pairs using the DeepSeekV3 API. After that, we explain our post-processing steps after obtaining the QA-pairs from the DeepSeekV3 API in Section 4.4. In Section 4.5, we provide an overview of our Dataset `CryptoQA`. At last, in Section 4.6 we postulate how our Dataset should be used in benchmarks.

4.1 Acquisition of data

Our first step requires collecting large-scale, high-quality data relevant to cryptography and related topic. This often requires scraping various online sources for data. These online sources contain structured as well as unstructured data, that needs to be cleaned and annotated to be used in downstream tasks. We used the online portal of the library of the University of Stuttgart to find high-quality sources. We then proceed to download all documents, including their chapters.

In the following, we explain our choices, which literature we considered for our approach and how we acquired those files.

Choice of Literature

The library of the University of Stuttgart provides a wide variety of online publications. As of 21.03.2025, there are over 2.549 books matching for cryptography. 1530 of those publications match for Springer as the publisher. We restricted ourselves to only using Springer Nature files to reduce the amount of different websites needed to be scraped. In addition to this, Springer Nature has a unified structure for all their webpages and fileservers, allowing an easy retrieval of documents in a large scale. As multilingual datasets come with a lot of their own problems, we exclude non-English publications. We furthermore removed “Hochschulschriften”, “Festschriften”, and “Zeitschrift/ Zeitung/ Serie” and offline publications to only include files in a certain structure.

At the end we were left with 1421 publications. The link to the online portal of the library with the selected filters can be found in the appendix A.

Acquisition of Links

Because the library does not provide an API to get the corresponding links and files, we performed web scraping to acquire the links. We used the Python library `BeautifulSoup4` to scrape the search result page of the library. We were able to obtain the title and the link of the online publications. To access all results not listed on the first page, we appended `&page=[pagenumber]` to the link to access all search results.

Acquisition from Springer Nature

The links to the publications of Springer Nature contain the DOI of each book and redirect to the website of the book of Springer Nature. There were some links that did not contain 10.1007, the DOI of Springer Nature. We did not follow those links and did not retrieve these files. Of the 1421 links, 32 links did not comply with this structure. One file was not valid anymore since the second edition of this book was released.

After that we obtained all possible documents using the following steps.

1. Using the scraped links to acquire the meta information about the books.
2. Downloading the file using the python `request` library.

Since downloading the files is an I/O bounded task, the retrieval could be accelerated using multi-threading in Python via the `joblib` library. With 10 in parallel running threads, this task in total took three hours to get all files.

In the end, we were left with 1388 documents. The documents are then put into folders given a description based on the information on the website of Springer. We thereby sort beforehand if a document is a textbook, book, conference proceedings or a reference work. Out of the 1388 files, two were reference works, 216 were books, 82 were textbooks, and 1088 were conference proceedings.

For each document, we also downloaded all its chapters separately which got their own DOI. Those chapters did not include table of contents or author indices. This worked for all documents except the two Reference works. Because of that, we did not further consider Reference works in our work as a source. Downloading the chapter directly allows us to not do any very complicated document splitting based on e.g., the bookmarks of the PDF files. In total, we downloaded 34998 chapters in total of which 1200 were from textbooks, 2855 were from books, and 30943 were from conference proceedings.

	number of files	number of chapter	number of pages
Textbook	82	1,200	29300
Book	216	2,855	59700
Conference proceedings	1,088	30,943	493153
Total	1,386	34,998	582153

Table 4.1: Overview of the scraped books with chapters

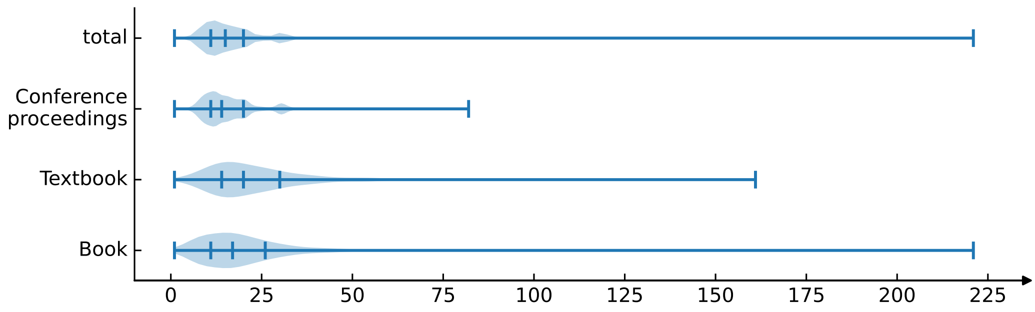

Figure 4.1: Distribution of the different chapter lengths

4.2 Preprocessing

The next step requires the preprocessing and cleaning of data. Since we downloaded all chapters separately, we do not need to split the documents with some heuristics like bookmarks. Furthermore, we do not need to filter out any unnecessary files that contain table of contents or author indices. However, as of 01.04.2025, the API of DeepSeek does not support files. Because of that, we need to perform text extraction first to be able to generate QA-pairs with DeepSeekV3.

Text extraction

Text extraction is a difficult task. Although Python3 libraries like *pypdf* provide a text extractor, due to their bad performance in extracting mathematical formulas, we used *MinerU*[49], also known as magic-pdf, for text extraction. *MinerU* is an AI-based text extraction software capable of extracting text from a PDF, while not extracting irrelevant text like page numbers as well as head- and footlines. The extracted text was formatted as Markdown. Tables used HTML syntax, and mathematical formulas were formatted using LaTeX. However, the text extraction did not work properly in some rare instances, resulting in incorrectly formatted LaTeX, which could not be rendered by Katex, the tool used by VS Code to render LaTeXin Markdown files.

However, the use of *MinerU* for text extraction resulted in an enormous computational demand compared to conventional approaches. To meet this computational demand, we used the bwUnicluster3.0 HPC infrastructure. bwUnicluster3.0 provides several CPU as well as GPU compute nodes.

We simultaneously used 2 GPU nodes with one NVIDIA H100 each to perform the text extraction. This task took more than 50 hours of GPU hours distributed over two weeks.

The extracted text of each file was then stored in a separate folder for each book.

4.3 Generation

As the generation of QA-pair in this scale cannot be done by a human annotator, we have to resort to an automated approach of generating QA-pairs. Using large language models (LLMs) to synthetically generate labeled dataset has gotten more attention recently[2]. We use DeepSeekV3 to generate the QA-pairs. In the remainder of this section we explain our model choices and our process to generate our pairs. By leveraging a Retrieval-Augmented Generation (RAG)-based approach, we want to ensure a high quality of our generated questions. We slightly deviate from the RAG pipeline by explicitly generating pairs for all books.

Selection of the model

DeepSeekV3 has shown that it can compete in English language understanding benchmarks with GPT4o and Claude-3.5-Sonnet and even outperform them in Chinese language understanding, code and even math benchmarks[9].

DeepSeekV3 is cheaper compared to similar products of OpenAI or anthropic. While DeepSeekR1 performs better in reasoning task than DeepSeekV3, it is twice as expensive as DeepSeekV3 at peak-hours and consumes more tokens due to its reasoning approaches.

As the GPT-4.1 series models were released too late for our work, we did not consider using them. Furthermore, DeepSeekV3 does not have any rate limits. DeepSeekV3-0329 is the latest version of DeepSeekV3.

DeepSeekV3 is a strong Mixture of Experts (MoE) model with 671 billion parameters[9]. It is based on the transformer architecture and uses Multi-Head Latent Attention. Its parameters, compared to models from OpenAI and anthropic are open source. It has a maximum context length of 64k, meaning that the number of tokens in input and output shall not exceed 64k. It's maximum output length is capped at 8k tokens, however the maximum output length was never an issue. One english characeter corresponds to 0.3 tokens for the DeepSeek API

We used `DeepSeek-V3-0329` to generate question answer pairs.

Querying

For each file, we used the following (Listing 4.1) query prefix to generate QA-pairs. During the querying process we slightly adjusted the prompt to fix some formatting issues. By explicitly setting the number of questions to generate to 25, we fixed the problem of generating not enough QA-pairs.

The DeepSeek API uses Markdown for formatting. Mathematical formulas are formatted using LaTeX-syntax. To ensure that we get enough questions per page, we counted the generated questions and generated more questions, until we get more than two questions per page or performed five rounds of queries. We set the maximum number of rounds to five to avoid not terminating due to incorrect formatting of the response object. Otherwise, we would have indefinitely generated QA-pairs.

We used the following(Line 2) prompt to generate more questions:

```
f"""Can you generate 25 question answer pairs for me about this specific text?
    Start Questions with **Q** and Answers with **A**. Separate Questions answer pairs with an
 empty line symbol. Also assign each questions a topic from one of the following types=
Applications, Cryptographic Protocols, Foundations, Implementation, Secret-Key Cryptography,
Public-Key-Cryptography,  Attacks and Cryptanalys, other Cyber Security, other Computer
Science.
    Write the topic below the answer in the following format. *topic*
    Furthermore classify each question as math if the answer is displaying calculations or
word if the answer is primarily an text with no calculations. Write the category below the
topic in the following format *category*
    {file_content}"""
```

Listing 4.1: the used prompt for QA generation.

```
"Can you generate more question answer pairs. Stick to the previously provided format."
```

Listing 4.2: the used prompt generating more QA-pairs.

We let the queries run in parallel to speed up the process. Furthermore, to make use of the discounted off-peak hours, we let the QA generation only run in between 18:30 and 2:30. Nevertheless, the total cost amounted to approximately 150€.

4.4 Postprocessing

After we finished generating the responses of DeepSeekV3, we proceeded to extract the question from the responses. In addition to this, it was necessary to remove faulty and duplicate questions. This section outlines the extraction of QA-pairs from the responses and the necessary cleaning steps.

Extraction of QA-pairs

The responses were all saved as separate JSON objects. After the generation process was done, we extracted the response text into a separate Markdown file. For 72 chapters not QA-pairs were generated as they exceeded the context length of the DeepSeekV3 API. For five additional chapters, no QA-pairs were generated as their length was so long that DeepSeekV3 prematurely stopped its response, without returning a QA-pair. Based on these markdown files, we were able to extract the QA pairs. Although we specified a format, slight differences occurred in the formatting of the output. However, as they occurred regularly, we could handle them with a slightly more complex regular expression.

Besides question and answer, we created the following columns. We used the same columns as Barman[6]:

- type: The type should indicate if the QA pair was rephrased or not. All generated QA pairs were assigned “original”

- category: The type should indicate if the answer contained calculations and mathematical reasoning or if the answer is primarily a text with no calculations. In the first case, the assigned category was "math", and in the latter "word".
- topic: Topic should indicate to which part of cryptography this QA pair belongs. The topics were taken from `eprint.iacr.org`. In addition to this, we assigned a topic "other cybersecurity" and "other Computer science" to label questions, which do not directly refer to cryptography.
- sources: This contains the DOI of the chapter

The fields topic and category were assigned by DeepSeekV3.

Filtering of Questions

In total, 1274971 question-answer pairs were generated from 1386 books with 34919 chapters. After removing exact duplicates, only 1274371 were left. 1273929 pairs were left after dropping all duplicate QA-pairs, where question and answer or question and source were identical. Duplicate questions or duplicate answers were not removed. Removing rows with unassigned values resulted in 1273490 remaining QA-pairs.

While we allow for duplicate questions, we did remove the most frequent questions, as those questions could only be answered if the source of the question was known. As a consequence, questions with more than 40 occurrences were removed. In addition to this, questions that related explicitly to the source were removed. Such questions were identified by certain keywords. The list of those of keywords can be found in the appendix.

After filtering out all such occurrences we were left with 1067787 pairs. After removing all pairs with an invalid category we were left with 1067690 QA-pairs.

	Book	Chapter	Chapter without QA-pairs	questions generated	avg questions per page
Textbook	82	1,200	34	34004	1.48
Book	216	2,855	38	71549	1.65
Proceedings	1,088	30,943	19	962137	2.13
Total	1,386	34,998	91	1067690	2.07

Table 4.2: Key statistics of the dataset

4.5 Overview

In this section, we provide an overview of the by DeepSeekV3 generated QA-pairs. We delve into each aspect of the dataset. An example of a QA-pair can be found in Figure 4.3 The answers to the questions are relatively short, spanning usually over only one to two sentences.

In total, 1067690 pairs were left after filtering. Although textbooks and made up 5.0% and books 10.2% of the total number of pages, they are a little bit underrepresented per page. This is due to more books and textbook files exceeding the context length of DeepSeekV3, and some changes that we implemented during the generation process.

In total over 2500 topics were assigned. Besides the given topics only one other topic, "Machine Learning" exceeded 1000 QA pairs. 90 % of QA-pairs were generated from Conference proceedings, 3.2 % were from Textbooks and 6.7 % were from Books.

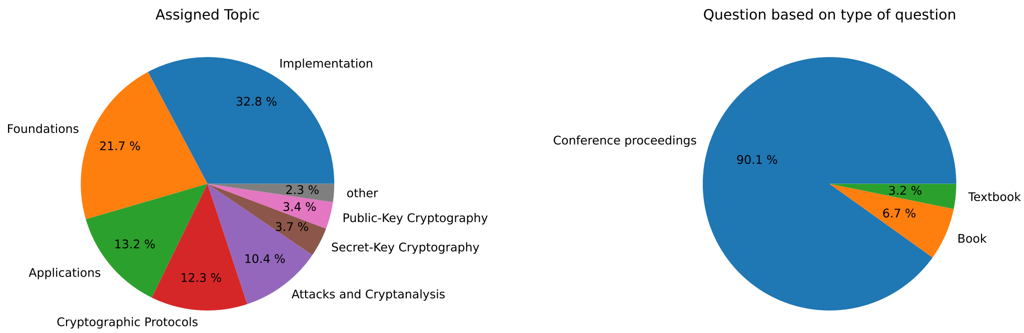

Figure 4.2: Distribution of the types of question

In total, 204616 QA-pairs were assigned the category "math", while 863074 were assigned the category "word".

Example of a QA-pair

Question:

```
What is the condition for \( G_1 \oplus G_2 \) to be cyclic if \( G_1 \) and \( G_2 \)
are finite cyclic groups?
```

What is the condition for $G_1 \oplus G_2$ to be cyclic if G_1 and G_2 are finite cyclic groups?

Answer:

```
\( G_1 \oplus G_2 \) is cyclic if and only if the orders of \( G_1 \) and \( G_2 \) are
 relatively prime (i.e., \(\gcd(o(G_1), o(G_2)) = 1\)).
```

$G_1 \oplus G_2$ is cyclic if and only if the orders of G_1 and G_2 are relatively prime (i.e., $\gcd(o(G_1), o(G_2)) = 1$).

type: original, **category:** word, **topic:** Foundations, **source:** 10.1007/978-3-031-21307-6_9

Figure 4.3: Example of a Question answer pair

4.6 Benchmark

To reliably and fairly evaluate LLMs, standardized benchmarks are needed. To ensure a fair comparison, we split `CryptoQA` into a train and test set with a ratio of 4 to 1. This splitting did not significantly impact the distribution of each category and topic. To avoid duplicate questions in the benchmark, we drop duplicate questions while still keeping the first occurrence of the question. In total, we removed 1255 QA-pairs resulting in benchmark with 212283 distinct questions.

Setup for evaluation

To fairly evaluate LLMs on our dataset, each run needs to use the same setup. As our answers are relatively short, the length of candidate answers should be restricted to at most 3 sentences.

Evaluation metric

To provide a useful benchmark, the evaluation metric needs to be chosen according to the task and the specific data. Since for both question categories, the answers are in a free-text form. We need to find a suitable metric for evaluating such answers. However, evaluating such answers in an ongoing problem far from being solved[38]. We propose to use BLEURT in a benchmark to evaluate LLMs.

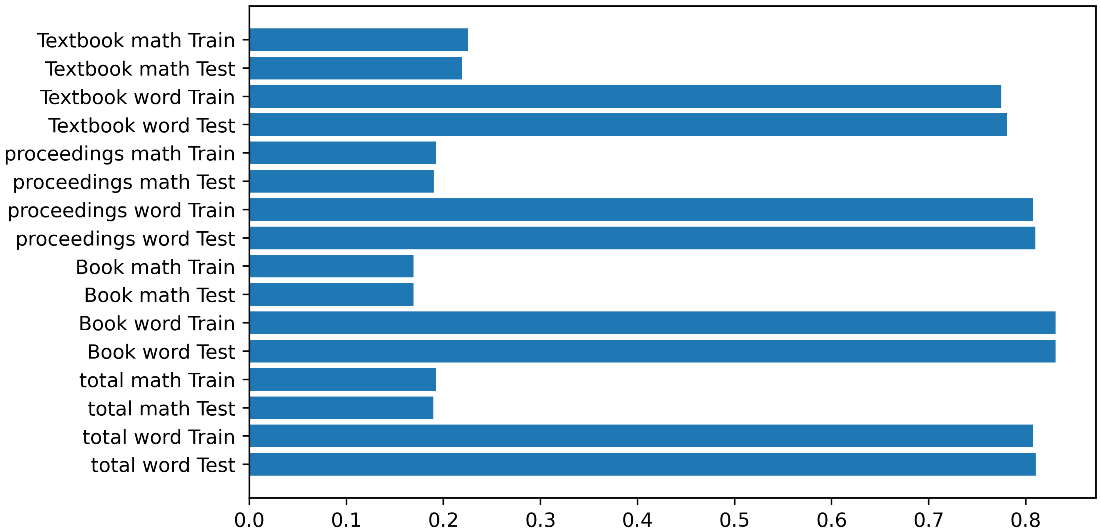

Figure 4.4: Percentages of book type and category in test and train data set

5 Evaluation

To assess the current capabilities of large language models (LLMs), we evaluate DeepSeekV3 and DeepSeekR1 with a focus on their understanding of cryptographic concepts. In Section 5.1, we explain our setup and the steps we took during the evaluation. In Section 5.2, we evaluate the results quantitatively and qualitatively.

5.1 Setup

In this section, we detail the experimental setup used to evaluate the cryptographic reasoning capabilities of two language models, DeepSeekV3 and DeepSeekR1, using the CryptoQA dataset. Our evaluation framework consists of three main components: the prompt design, the answer generation, and the metric-based assessment.

We begin by outlining the prompting strategies employed to elicit responses from each model. Specifically, we describe how questions from the CryptoQA dataset are formatted and presented to the models, and how the responses to the questions are collected. The prompts are carefully crafted to ensure consistency across models and to minimize prompt-induced variance in performance.

Following the prompt design, we explain the evaluation metrics used to quantify the performance of the models. We employ different evaluation metrics, which are commonly used in natural language understanding tasks.

This setup allows for a rigorous and reproducible comparison of DeepSeekV3 and DeepSeekR1 in the domain of cryptography, offering insights into the models' relative strengths and limitations.

5.1.1 Generation of answers

We evaluate DeepSeekV3 and DeepSeekR1 on the CryptoQA dataset. We use the DeepSeek API to generate the answers from DeepSeek-V3-0324 and DeepSeek-R1-0528, each with a temperature of 1. The following prompt was used to generate the results:

```
f"""Can you answer the following question for me. Write at most three sentences but try to
keep it shorter:
        {question}"""
```

Listing 5.1: the used prompt for QA generation.

We used 200 threads running in parallel to accelerate the generation of the answers. It took roughly 8 hours of querying to generate the answers. We used the bwUnicluster3.0 to compute our results. We made use of the off-peak-hours discount of DeepSeek like in the generation of question answer

(QA)-pairs. While during the off-peak hours the price per-token was the same, the generation of reasoning tokens by DeepSeekR1 led to a significantly higher consumption and billing of output tokens. While we amended to use preferably fewer sentences, the answers generated by DeepSeekV3 and DeepSeekR1 were often longer than our reference answers. The average answer from CryptoQA contained 16 words while the answers of DeepSeekV3 and DeepSeeekR1 contained 46 and 49 words per answer.

5.1.2 Used Metrics

We used BLEU, ROUGE, BERTscore, and BLEURT for evaluation of the answers:

- **BLEU** is a metric originally developed for machine translation. It evaluates how closely a generated text matches one or more reference texts[35]. BLEU measures n-gram precision, i.e., how many contiguous sequences of n words in the candidate output also appear in the reference text. A brevity penalty is applied to discourage overly short responses[35]. BLEU is simple, widely used, and works well for structured text like translations. However, it does not consider semantic meaning and penalizes valid paraphrases that use different wording[44]. In this work, we use the default configuration of N = 4 and equal weights[35]. The scores were calculated using the Python library nltk with its tokenizer punkt.
- **ROUGE** is a metric primarily used for evaluating summarization tasks by comparing overlaps between the generated and reference summaries[27]. It focuses on recall rather than precision, though some variants consider both. ROUGE has several variants measuring different aspects. ROUGE-N measures the n-gram recall e.g., ROUGE-1 for unigrams, ROUGE-2 for bigrams. ROUGE-L is based on the Longest Common Subsequence between the candidate and reference text. ROUGE is good for measuring coverage of reference content [27]. However, like BLEU, it is based on surface form overlaps and doesn't handle paraphrasing or semantics well. In this work, we calculated ROUGE-1, ROUGE-2, ROUGE-L and ROUGELsum. The computations were done by using the Python library evaluate.
- **BERTscore** measures semantic similarities between texts using deep contextual embeddings from BERT or similar transformer models. Each word/token in the candidate and the reference text is embedded using a pre-trained language model (e.g., BERT). Then, the cosine similarity is calculated between tokens, and the best matches are aggregated to produce a final score (precision, recall, and F1 variants). BERTscore captures semantic similarities beyond surface-level matching and is robust to paraphrasing. However, it is computationally demanding and sensitive to the choice of pre-trained model and tokenization. In this work, we use the Python library bert-score with the options `lang="eng-sci"` and `model_type="microsoft/deberta-xlarge-mnli"`, which is the recommended model by the authors of the library. We calculate the BERT recall score, the BERT precision score, and the BERT F1 score.
- **BLEURT** is a learned evaluation metric that combines contextual embeddings with human-annotated data to predict the quality of generated text[41]. BLEURT is fine-tuned on a dataset of human ratings of text quality (e.g., WMT). It uses BERT-like models to encode both reference and candidate, and predicts a quality score based on their relationship. BLEURT has

a strong correlation with human judgment in evaluating text generation and can handle fluency, adequacy, and semantic similarity well[41]. Here, we use the reference implementation by google-research with the BLEURT-20 version.

We used the bwUnicluster3.0 to compute the individual scores for each answer.

To evaluate the correctness of answers with the above-described metrics, a threshold needs to be chosen. In prior works evaluating free text answers with BERTscore and BLEURT, this threshold has been set to 0.2[50]. However, as this threshold did not work for us, we opted to use the threshold of 0.6 for the evaluation with BLEURT.

5.2 Results

In this section, we present and evaluate the results of our comprehensive evaluation of DeepSeekV3 and DeepSeekR1 on the CryptoQA dataset. A total of nine distinct evaluation metrics were employed to assess model performance from both surface-level and semantic-based metrics. These include lexical overlap measures (e.g., BLEU, ROUGE), embedding-based similarity metrics (e.g., BERTscore), and learned metrics (e.g., BLEURT), providing a multifaceted view of the models' capabilities. Our evaluation is further stratified across three dimensions: (i) Topic — representing distinct areas of cryptographic knowledge, (ii) Book Type — corresponding to the source and style of question derivation, and (iii) Category — denoting the classification of question types. An overview of the score distributions across these dimensions is visualized in Figure 5.1, illustrating the variation in model performance as measured by different metrics. For every metric, we compare the performance of DeepSeekV3 and DeepSeekR1 across all evaluation categories. Special attention is paid to identifying systematic patterns or discrepancies.

In the second part of this chapter, we assess the performance and effectiveness of our proposed evaluation metric, with a particular focus on its alignment with human judgment.

5.2.1 Quantitative results and discussion

Surprisingly, DeepSeekV3 outperformed DeepSeekR1 across nearly all evaluation metrics. As seen in Figure 5.1, the scores of DeepSeekV3 are equal if not better than the scores of DeepSeekR1. This initially surprising result required a closer examination of the evaluation setup and the metric behavior. An overview of the average values can be found in Table 5.1. An overview of the scores for each category and book type can be found in Table B.1. The scores for each topic can be found in Table B.2.

Upon further analysis, this performance discrepancy can be partially attributed to the origin of the ground-truth answers, which were themselves generated by DeepSeekV3. As a consequence, metrics that rely heavily on semantic or embedding-based similarity, such as BERTscore and BLEURT, are likely to favor responses that are stylistically and semantically aligned with those ground-truth references. This inherent alignment introduces an inherent bias in favor of DeepSeekV3, particularly in metrics that go beyond surface-level token overlap and assess deeper representational similarity.

Thus, while the higher scores of DeepSeekV3 reflect its ability to reproduce the reference answers effectively, they must be interpreted with caution, especially when comparing against DeepSeekR1 and other models. This observation underscores the importance of neutral or human-authored ground truth in fair and unbiased model evaluation.

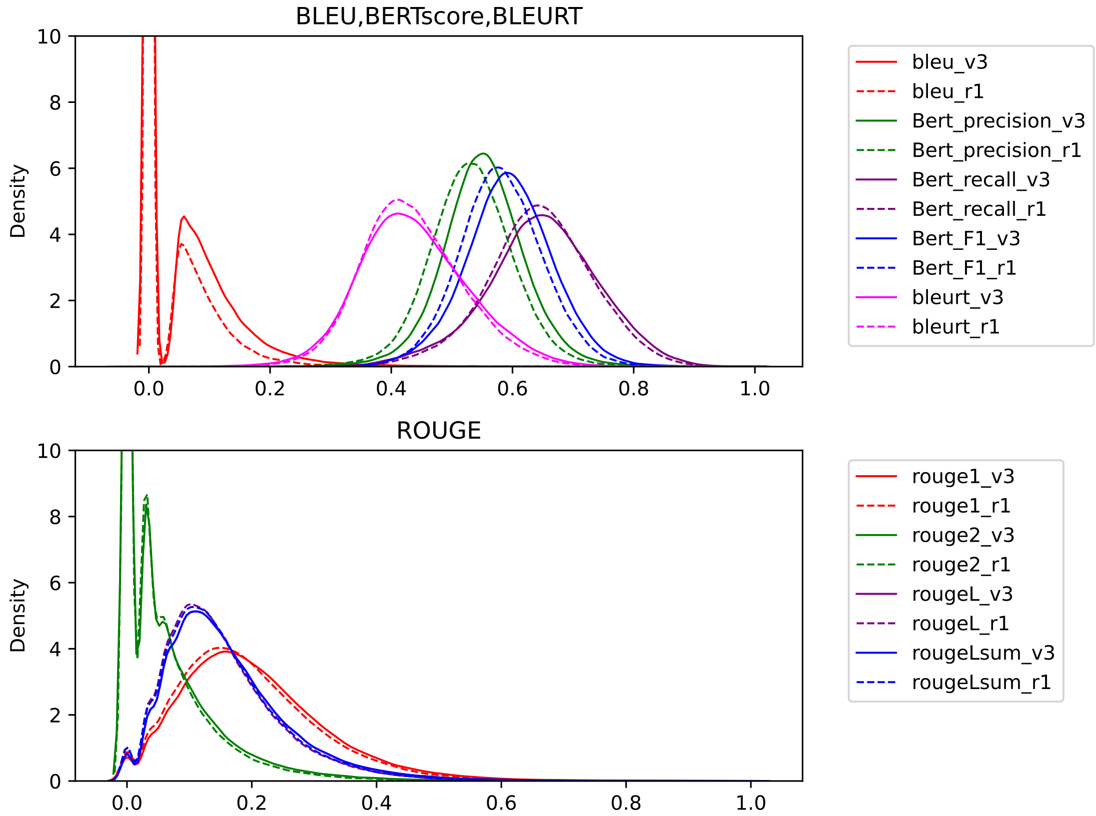

Figure 5.1: Distributions of the scores over the whole dataset

The performance trends observed in the overall evaluation persist when the results are disaggregated by book type. Specifically, DeepSeekV3 consistently outperforms DeepSeekR1 across most metrics, regardless of the type of source material from which the questions were derived. This pattern reinforces the earlier observation of metric bias favoring the model that generated the reference answers.

Notably, the average scores for questions derived from textbooks and general books are higher compared to the overall average across all book types. This suggests that questions from these sources may be more aligned with the language models' training data or more amenable to precise generation. However, despite this difference in absolute scores, the distributional shape of scores across all book types closely mirrors that of the Conference Proceedings subset. This resemblance implies that while average performance levels vary, the overall structure of performance variability remains relatively stable across book types.

A detailed visualization of the score distributions grouped by book type and aggregated across the dataset is provided in Figure 5.3 and Figure 5.4, enabling closer inspection of these patterns.

	mean	std	min	25%	50%	75%	max
Bert_F1_r1	0.5814	0.0699	0.2411	0.5358	0.5799	0.6261	1.0000
Bert_F1_v3	0.5947	0.0727	0.2449	0.5487	0.5945	0.6414	1.0000
Bert_precision_r1	0.5326	0.0695	0.1884	0.4883	0.5317	0.5761	1.0000
Bert_precision_v3	0.5525	0.0690	0.1971	0.5099	0.5515	0.5943	1.0000
Bert_recall_r1	0.6454	0.0872	0.1998	0.5905	0.6456	0.7024	1.0000
Bert_recall_v3	0.6489	0.0934	0.2217	0.5916	0.6503	0.7107	1.0000
bleu_r1	0.0084	0.0333	0.0000	0.0000	0.0000	0.0000	1.0000
bleu_v3	0.0113	0.0405	0.0000	0.0000	0.0000	0.0000	1.0000
bleurt_r1	0.4345	0.0842	-0.0322	0.3775	0.4284	0.4870	0.9926
bleurt_v3	0.4381	0.0905	-0.0435	0.3763	0.4316	0.4956	0.9799
rouge1_r1	0.1932	0.1113	0.0000	0.1143	0.1772	0.2545	1.0000
rouge1_v3	0.2039	0.1157	0.0000	0.1224	0.1875	0.2667	1.0000
rouge2_r1	0.0577	0.0765	0.0000	0.0000	0.0323	0.0842	1.0000
rouge2_v3	0.0644	0.0842	0.0000	0.0000	0.0351	0.0938	1.0000
rougeL_r1	0.1531	0.0952	0.0000	0.0870	0.1351	0.2000	1.0000
rougeL_v3	0.1625	0.1008	0.0000	0.0930	0.1429	0.2105	1.0000
rougeLsum_r1	0.1548	0.0955	0.0000	0.0889	0.1370	0.2000	1.0000
rougeLsum_v3	0.1625	0.1005	0.0000	0.0930	0.1429	0.2105	1.0000

Table 5.1: Statistical numbers of the evaluation

When we further divide the results by question category, we observe notable differences in all distributions, particularly between the "math" and "word" categories. The "math" questions, typically requiring more symbolic reasoning and precision, yielded lower BERTscore and BLEURT scores on average, likely due to the semantic embedding models being less suited for capturing correctness in mathematical expressions. In contrast, "word"-based questions, which are more language-focused, achieved higher scores in semantic-similarity-based metrics.

Interestingly, for metrics based on n-gram overlap, this behavior is exactly the opposite. Especially for the BLEU and ROUGE-2 metric, DeepSeekV3 and DeepSeekR1 achieved more than 20% lower scores in the category word compared to the category math. On ROUGE-1, ROUGE-L and ROUGE-Lsum, DeepSeekV3 and DeepSeekR1 achieved higher scores on the "math"-category compared to "word", albeit with a smaller margin.

When disaggregating the results by the topic of the questions, we can also notice differences across the different evaluation metrics. We grouped topics that are not a valid category on the cryptology eprint archive into the topic "other". While there are larger differences between the average scores of individual topics for lexical-match-based metrics like BLEU and ROUGE, the difference between the topics with the highest score and the lowest score is quite small for semantic evaluation metrics like BERTscore and BLEURT. On the topic "foundations" the scores were often the highest, while on the topic "Attacks and Cryptanalysis" the scores were, with the exception of the BLEURT metric, the lowest. The topic with the highest and lowest scores were usually the same for DeepSeekV3 and DeepSeekR1, with the exception of the BERT F1 and BERT Precision metric.

BLEU

Across all evaluation settings, BLEU scores were consistently low for both DeepSeekV3 and DeepSeekR1. This outcome is primarily attributed to the scarcity of 4-gram overlaps between the model-generated answers and the reference answers, a core component in the computation of BLEU scores. Since BLEU relies on exact n-gram matches, its effectiveness diminishes in scenarios where semantically correct answers are expressed using alternate phrasing or where linguistic variability is high.

Quantitatively, only 10.47% of the answers generated by DeepSeekV3 and 8.65% of those generated by DeepSeekR1 exhibited a 4-gram overlap with the corresponding reference answers across the entire dataset. These low percentages reflect the limited lexical similarity between candidate and reference responses, highlighting BLEU's limitations in evaluating tasks that permit paraphrasing or multiple correct formulations.

However, a more granular analysis reveals substantial variation in 4-gram overlap across different book types. For instance, within the 'Conference Proceedings' subset, only 9.13% of DeepSeekV3's answers and 7.56% of DeepSeekR1's answers contained 4-gram matches with the reference. This low overlap likely stems from the more technical and variable language used in conference-style academic writing, where answers may differ significantly in surface form.

In contrast, the "Textbook" book type, which is presumably characterized by more structured or pedagogical content, yielded markedly higher overlap rates. Specifically, 32.05% of DeepSeekV3's answers and 27.77% of DeepSeekR1's answers from this subset exhibited 4-gram overlap with the reference answers. This suggests that questions derived from source books may encourage more predictable, formulaic responses that align closely with reference formulations.

The BLEU score exhibited the largest relative difference between the individual topics. The score of the topic with the largest score for both models "Foundations" was more than 2.5 times larger than for the topic with the lowest score "Attacks and Cryptanalysis".

These findings emphasize the sensitivity of BLEU to the stylistic and lexical properties of the data source, and underscore the need to complement it with semantic-aware metrics (e.g., BERTscore, BLEURT) when evaluating open-ended or domain-specific language generation tasks.

BLEURT

The evaluation using the BLEURT metric, which incorporates contextual embeddings and is fine-tuned to align with human judgment, reveals a small difference in performance between the two models under study. The average BLEURT score for DeepSeekV3 is 0.4391, while DeepSeekR1 achieves a nearly comparable score of 0.4344. This results in a relative difference of only 0.83% between the two models.

Like BLEU, the BLEURT scores vary significantly between different book types. The pattern, in which the models scored the best on questions from the textbooks and the worst on questions from conference proceedings, prevails. DeepSeekV3 achieved higher scores than DeepSeekR1 in every book type.

While the scores disaggregated over the book type vary significantly, this is not the case for the categories. The average BLEURT score of DeepSeekV3 over questions of the category “word” is only 7.20% higher than the average scores over the category “math”. For DeepSeekR1, this relative difference is with 5.35% even smaller. DeepSeekR1 achieved over the category math an average score of 0.4164, which is slightly better than the score of DeepSeekV3, which only achieved a score of 0.4140.

For the BLEURT metric, each topic achieved similar scores. The topic with the highest score “other” was only 6.81% higher for DeepSeekR1 and 7.3% higher for DeepSeekV3 than “Implementation”, the topic with the lowest scores. BLEURT was also the only metric where “Attacks and Cryptanalysis” was not the topic with the lowest score across each metric.

Such a narrow margin suggests that, from the perspective of BLEURT, which captures not only lexical and semantic similarity but also fluency and adequacy, the two models perform similarly in terms of producing high-quality answers. Unlike surface-level metrics (e.g., BLEU), BLEURT accounts for paraphrasing and nuanced variations in phrasing. Therefore, these small differences in average scores indicate that DeepSeekR1 is, in many cases, able to generate semantically equivalent and contextually appropriate responses that are nearly indistinguishable from those of DeepSeekV3.

However, the overall similarity in BLEURT scores stands in contrast to the larger discrepancies observed in other metrics, particularly those that are sensitive to n-gram overlap or lexical structure. This further underscores BLEURT’s robustness and its value as a metric when comparing models in tasks, where linguistic variability and paraphrasing are expected.

While DeepSeekV3 holds a slight edge in BLEURT scoring, the magnitude of the differences is modest, indicating comparable semantic-output quality across the two models within the evaluated cryptographic domain.

BERTscore

Compared to BLEURT, the models achieved higher scores on the BERT F1 metric. The average score of DeepSeekV3 was 0.5947, while DeepSeekR1 achieved only a score of 0.5814, resulting in a relative difference of 2,29%.

Similar to the results of BLEURT, the average scores on questions from “Textbooks” were 0.6478 for DeepSeekV3 and 0.6326 for DeepSeekR1, significantly higher than the scores over the category “Conference Proceedings” with 0.5901 for DeepSeekV3 and 0.5771 for DeepSeekR1. The average scores over the category “Books” were with scores of 0.6299 and 0.6142 slightly lower than the scores of the textbooks.

DeepSeekR1 achieved a 4.90 % higher score on questions of the category “word” compared to the category “math”. DeepSeekV3’s score was 6.40% higher on “word”-questions compared to “math”questions.

For the BERT recall and BERT precision score, the for BERT F1 described patterns persist. DeepSeekV3 outperformed DeepSeekR1 in all book types and categories. The scores over the book type “Textbook” were the highest, while the scores on “Conference proceedings” were the lowest. Also, the average scores were lower in the category “math” than in the category “word”.

The differences in the scores across all metrics were quite small, with a maximum of 5.23% . However, the BERT F1 and BERT Precision metrics were the only metrics where the topics with the highest score differed by the model. While DeepSeekR1 achieved the highest scores on the "Foundations" metrics, DeepSeekV3 scored the highest on the "Application" topic. For the BERT Recall metric, both models achieved the highest scores on "other" topics.

The R_{BERT} scores were always higher than the P_{BERT} scores across all categories and book types and models. However, this is not unexpected as the candidate answers were generally longer than the reference answers leading to higher recall scores compared to the precision scores. The $F1_{BERT}$ scores, the harmonic mean of R_{BERT} and P_{BERT}, are between the R_{BERT} and P_{BERT} scores.

That DeepSeekV3 is outperforming DeepSeekR1 in all of BERTscore's metrics is not surprising, as the scores are based on the sum of cosine similarities between the tokens of the reference answer and the candidate answer[54]. Since the reference answers are generated by DeepSeekV3, these metrics are inherently biased towards DeepSeekV3.

ROUGE-1

The higher average score for DeepSeekV3 with 0.2039 persists compared to DeepSeekR1 with only 0.1932 at ROUGE-1. This behavior is present for every book type and category. While scores over each book type followed the pattern of previously described scores, the scores over the category "word" were 4.38 % lower than the scores of the category "word" for DeepSeekV3 and 5.99 % lower for DeepSeekR1.

Compared to the semantic-based evaluation metrics, the relative differences between the scores over each topic are larger. On questions of topic "Foundations" both models achieved at least 30% higher scores compared to the topic with the lowest average score "Attacks and Cryptanalysis".

ROUGE-2

The average ROUGE-2 score for DeepSeekV3 was with 0.0645 11.61 % higher than the score of DeepSeekR1, which only achieved a score of 0.0577. The average ROUGE-2 scores exhibit the same behavior across different book types and categories as the ROUGE-1 scores. However, the differences between the scores divided into book types are much larger. While the relative difference of the ROUGE-1 score for DeepSeekV3 between the category "math" and "word" was -4.38%, the relative difference of the ROUGE-2 scores is more than 70%. These differences occur in the scores of DeepSeekR1 as well.

The same is true for the differences in scores between the different topics. While for ROUGE-1 and ROUGE-2 the topic with the highest and lowest scores were the same, the DeepSeekV3 relative difference between the topics increased from 33% to over 79%.

ROUGE-L and ROUGE-Lsum

As there was often only one reference sentence, the ROUGE-L and ROUGE-Lsum values are often identical. The ROUGE-Lsum score for DeepSeekR1 was just 1.14% higher than the ROUGE-L score. For DeepSeekV3, this difference was even smaller with just 0.002%.

However, the differences between each category, topic, and book type, like in ROUGE-2, remain. For both metrics and models, the relative differences between the different book types were just below 70The relative difference of the categories “math” and “word” in scores is just about 10%.

However, the maximum relative differences between each topic was at 35% for both metrics and models.

Correlation between the different scores

To evaluate how well high scores in one metric correspond to high scores in other metrics, we calculated the Spearman correlation coefficient. The individual scores can be found in Figure 5.2. The Spearman correlation coefficient is a rank-based metric to evaluate, how monotonously two variables are related[11]. All scores exhibit at least a moderate correlation between each other. The correlation is especially high for closely related metrics. The correlation between the same metrics but in different models is very high.

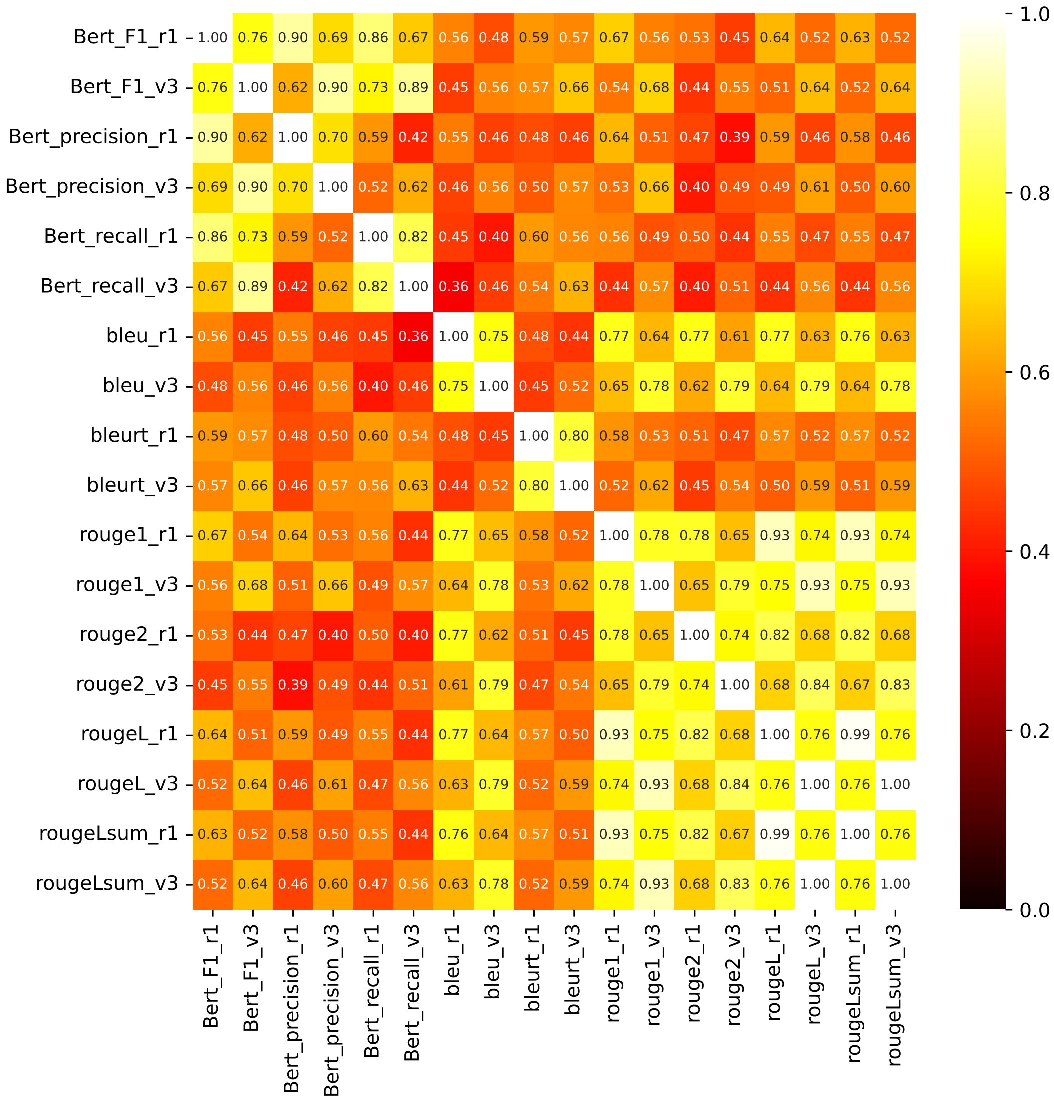

Figure 5.2: Spearman correlation coefficient

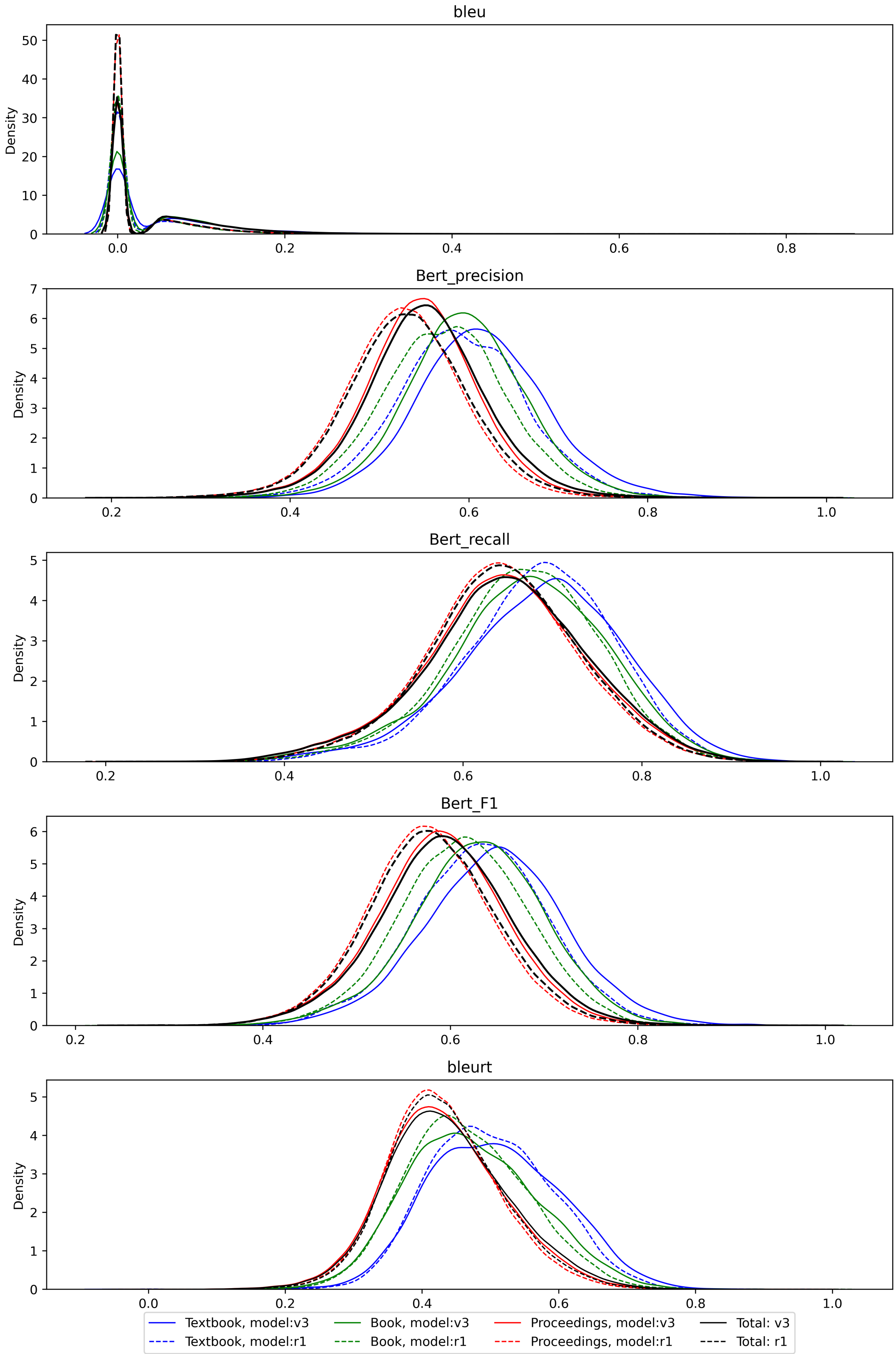

Figure 5.3: Distributions of the scores per book type Part 1

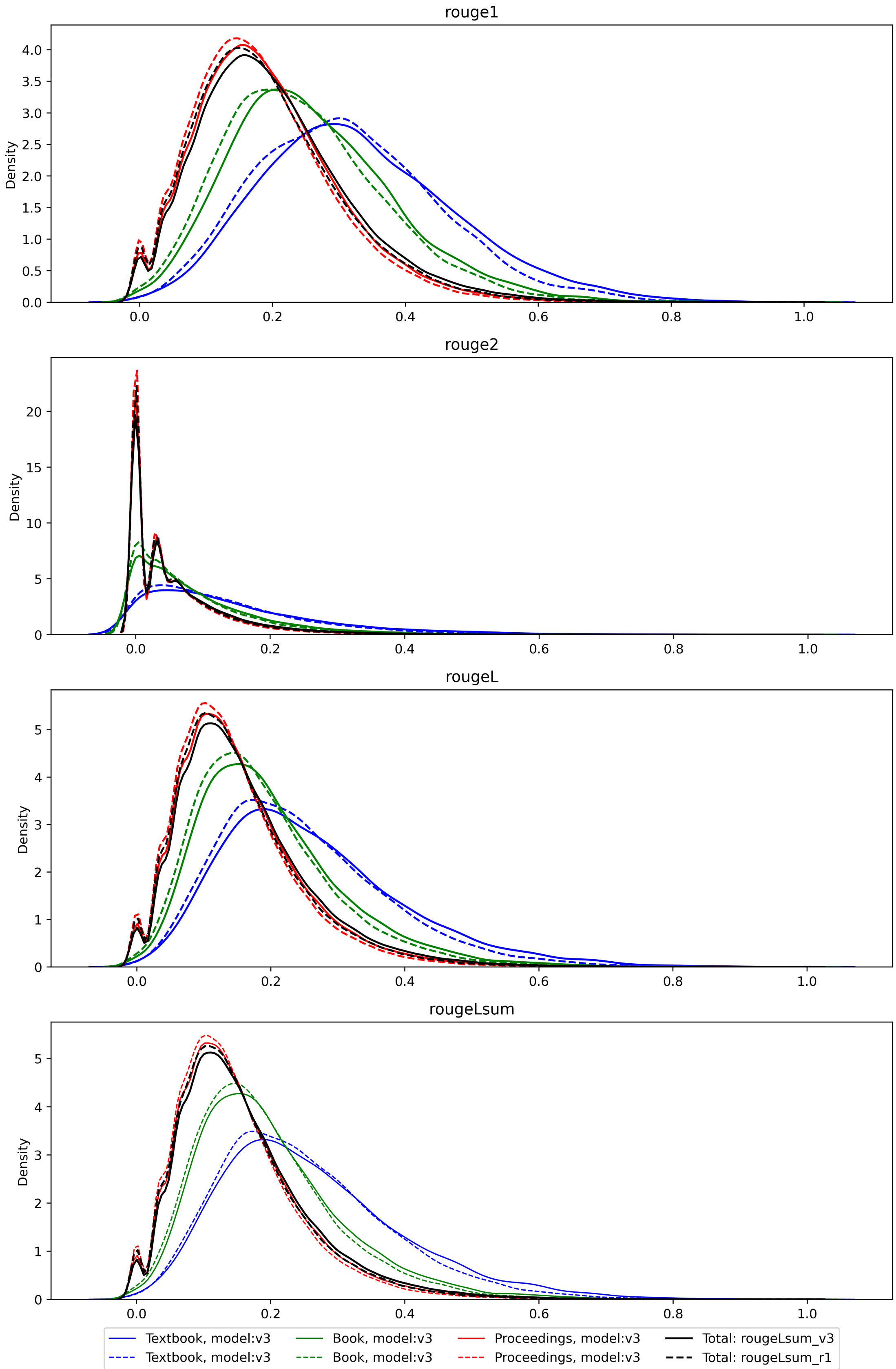

Figure 5.4: Distributions of the scores per book type Part 2

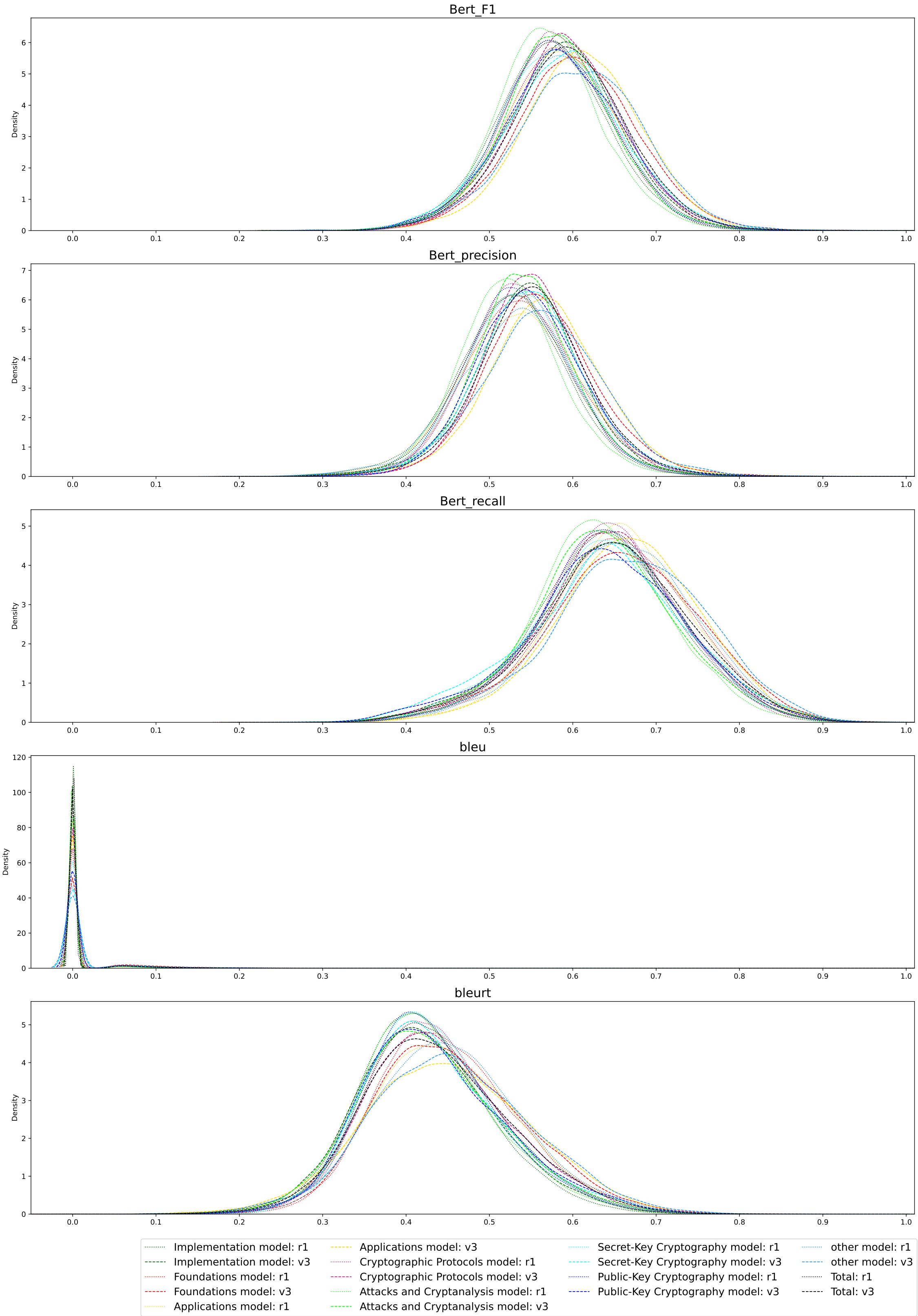

Figure 5.5: Distributions of the scores per topic Part 1

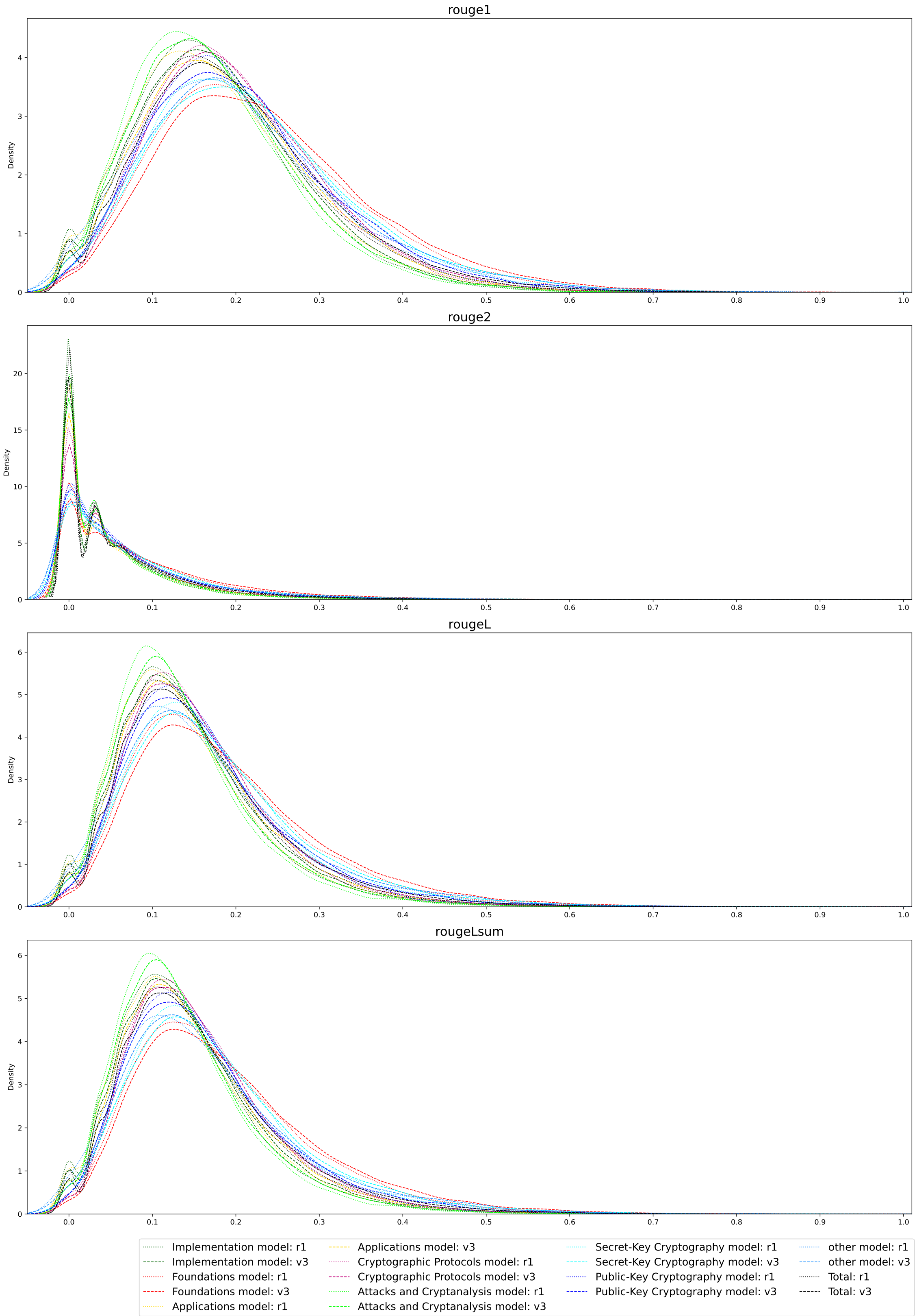

Figure 5.6: Distributions of the scores per topic Part 2

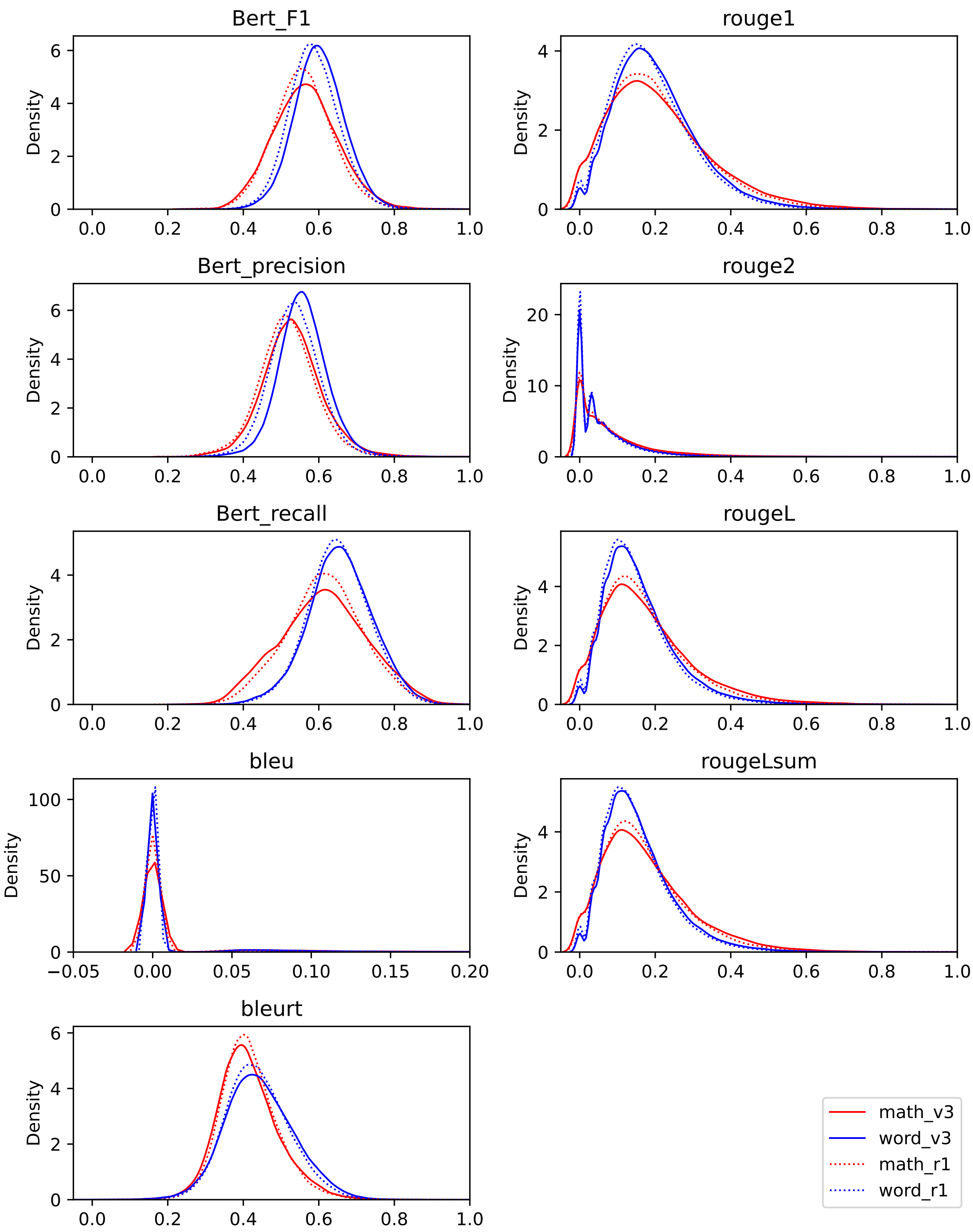

Figure 5.7: Distributions of the scores per category

5.2.2 Qualitative evaluation

While the preceding sections focused on the quantitative evaluation metrics applied to the model outputs, this section shifts attention to a qualitative analysis aimed at assessing how well these metrics align with human-perceived answer correctness and quality.

For our proposed evaluation metric BLEURT, we examine to what extent high metric scores correspond to answers that are factually accurate, contextually appropriate, and semantically correct. This analysis is crucial for understanding the validity and interpretability of automated evaluation methods, particularly in specialized domains such as cryptography, where subtle errors may compromise the overall correctness of an answer.

We conduct this examination by manually reviewing a representative subset of answers that received both high and low scores according BLEURT. Through this process, we assess whether the metric's scoring behavior reflects meaningful distinctions in answer quality, or whether it is influenced by superficial similarities that may not correspond to true correctness.

In addition to evaluating metric alignment with answer quality, we also identify and analyze common failure modes exhibited by the evaluated models.

This qualitative investigation provides a more nuanced understanding of the strengths and limitations of both the models and the evaluation metrics. It further informs our recommendations for metric selection and highlights areas for future improvement in model training and prompting strategies.

Qualitative results and discussion

A substantial number of questions in the dataset required contextual information from the source document for an accurate resolution, as they contained references to specific phrases or statements embedded within the original text. Without access to this surrounding context, it becomes challenging for models to generate precise or fully grounded responses.

```
question: How does the Link algorithm work?

answer: It checks if two signatures share the same deterministic linking tag \([f]_l = \
mathcal{E}([d]_{l-t}, [x]_t)\) for the same event/time.

answer\_v3: The Link algorithm in UpRank identifies and prioritizes the most relevant links
for a given query by analyzing factors like anchor text, source authority, and contextual
relevance. It uses machine learning to assess link quality and ranks them based on their
potential to improve search performance. The goal is to surface the most valuable links for
users efficiently.
answer\_r1: The Link algorithm, often referring to PageRank, ranks web pages by analyzing
hyperlink structures:
1. Pages gain importance from incoming links, treating each as a "vote."
2. Votes from highly linked pages carry more weight, creating a recursive importance score.
3. This process models web surfing behavior, prioritizing pages with robust, quality backlinks
.
```

```
Scores: bleu_v3: 9.3361e-232, bleu_r1: 0.0, bleurt_v3: 0.34550, bleurt_r1: 0.3573,
Bert_precision_r1: 0.4480, Bert_recall_r1: 0.4260, Bert_F1_r1: 0.4367, Bert_precision_v3:
0.5122, Bert_recall_v3: 0.4348, Bert_F1_v3: 0.4703
```

Listing 5.2: Example of wrong context

This lack of contextual grounding frequently resulted in hallucinations, where the models generated answers that diverged from the initial intent of the question. The most common errors in such cases involved the insertion of incorrect numerical values or the production of entirely unrelated answers that bore little relevance to the original question.

LLMs often struggle to answer questions that reference specific phrases from source documents because they lack access to the full contextual information unless it is explicitly included in the prompt. These models are trained to generate likely responses based on general language patterns, not to perform document-specific retrieval or reasoning. As a result, when questions rely on local textual cues or intra-document references, LLMs cannot resolve them accurately without additional context or retrieval mechanisms. This limitation is particularly prominent in technical domains like cryptography, where precision and context fidelity are critical.

Interestingly, in some instances, DeepSeekV3 delivered a correct answer, despite it missing the context necessary to answer the question, suggesting stronger generalization or memorization capabilities. In contrast, DeepSeekR1 occasionally acknowledged its limitations by explicitly stating that it could not answer the question given the available context of the input. This divergence highlights differences in model behaviour, particularly in how they handle uncertainty and incomplete contextual information.

```
question: What percentage of respondents in EEE 422 were 'very interested' in poetry and
literature?

answer: 22\% of EEE 422 respondents reported being 'very interested' in poetry and literature.

answer\_v3: In EEE 422, 15% of respondents reported being 'very interested' in poetry and
literature.
answer\_r1: In EEE 422, 15% of respondents reported being 'very interested' in poetry and
literature.
scores: bleu_v3: 0.6263, bleu_r1: 0.6263, bleurt_v3: 0.8334, bleurt_r1: 0.8334,
Bert_precision_r1: 0.9227, Bert_recall_r1: 0.9364, Bert_F1_r1: 0.9295, Bert_precision_v3:
0.9227, Bert_recall_v3: 0.9363, Bert_F1_v3: 0.9295
```

Listing 5.3: Example of wrong context

BLEU

More than half of the QA-pairs do not contain a 4-gram overlap between the reference answer and the generated answer. Many answers don't even contain a 2-gram overlap. Due to implementation details of the library[1], this leads to BLEU values between the order of 10^{-200} and 10^{-70}. Only

[1] If there is no 4-gram overlap between the reference answer and the candidate answer, the library has to compute log(0), which is undefined.

8.56 % responses generated by DeepSeekR1 and 10.47 % of responses generated by DeepSeekV3 had a 4-gram overlap. For example, in the following question, the BLEU score for DeepSeekR1 is extremely low, although the answer is very close to the reference answer.

```
question: Show that if one-way functions exist, then \(\mathsf{P} \neq \mathsf{NP}\).

answer: One-way functions imply the existence of problems in \(\mathsf{NP}\) (e.g., inverting
the function) that are not in \(\mathsf{P}\) (since inversion is hard). Thus, \(\mathsf{P} \
neq \mathsf{NP}\).

answer\_v3: If one-way functions exist, then there are problems in \(\mathsf{NP}\) (e.g.,
inverting the function) that cannot be solved in polynomial time, implying \(\mathsf{P} \neq \
mathsf{NP}\). This is because one-way functions are efficiently computable but hard to invert,
 which contradicts \(\mathsf{P} = \mathsf{NP}\) since \(\mathsf{NP}\)-hardness would imply
efficient invertibility. Thus, \(\mathsf{P} \neq \mathsf{NP}\) is a necessary condition for
one-way functions to exist.

answer\_r1: If one-way functions exist, they are computable in polynomial time but not
invertible in polynomial time by any probabilistic algorithm for random inputs. This implies
the existence of a problem in \(\mathsf{NP}\) (inverting the function) that is not solvable in
 polynomial time. Therefore, \(\mathsf{P} \neq \mathsf{NP}\).

Scores: bleu_v3: 0.1526, bleu_r1: 3.0100e-78, bleurt_v3: 0.5989, bleurt_r1: 0.6065,
Bert_precision_r1: 0.6578, Bert_recall_r1: 0.7449, Bert_F1_r1: 0.6987, Bert_precision_v3:
0.6120, Bert_recall_v3: 0.7082, Bert_F1_v3: 0.6566
```

Listing 5.4: Example of a low BLEU score

While the BLEU score for the answer generated by DeepSeekV3 is 0.1526, the score for DeepSeekR1 is $3.01 \cdot 10^{-78}$, indicating there is no 4-gram in both the reference answer and the candidate answer.

BLEURT

To evaluate the scores, we view five samples for 10 intervals of the same length. We do this for both the responses generated by DeepSeekV3 and DeepSeekR1.

While in general the similarity between the answers and correctness of the candidate answers increased, there are still some issues with BLEURT. Especially when viewing the lower scores, there were a lot of occurrences where the answers generated by DeepSeekV3 and DeepSeekR1 diverged from the initial intent of the question. Answers of DeepSeekV3 and DeepSeekR1, which scored lower than 0.4 were, with the exception of one answer by DeepSeekV3, incorrect, albeit half of the errors were due to missing contexts. In between scores of 0.4 and 0.6, there was a mix of correct and wrong answers. Answers from models that were rated better than 0.6 were usually correct. However, questions that asked for concrete numbers were the exception, like Listing 5.3.

If we would apply our threshold, only 4.71% of all answers of DeepSeekV3 and 3.42 would be considered correct. However, due to the differences across book types 18 % of answers from DeepSeekV3 and 14 % answers by DeepSeekR1 to question from textbooks are exceeding our threshold. For questions from Book, 11% of answers by DeepSeekV3 and 7.9 % of answers by DeepSeekR1 are exceeding this treshold.

6 Discussion and Outlook

The results of our evaluation demonstrate that while large language models such as DeepSeekV3 and DeepSeekR1 exhibit promising language understanding capabilities, they are not yet sufficiently reliable for complex, domain-specific tasks in cryptography. Our findings reveal several core limitations that hinder their effective deployment in this field.

6.1 Limitations of current LLMs

First, both quantitative and qualitative analyses show that the models frequently struggle with context-dependent questions, particularly those that reference specific phrases or sections from source materials. In the absence of direct access to the surrounding textual context, the models often generate hallucinated content, including incorrect numerical values or irrelevant explanations. This lack of grounding is particularly problematic in cryptography, where accuracy and precision are essential.

Moreover, although DeepSeekV3 generally outperforms DeepSeekR1 in most metrics, this advantage can be partially attributed to the bias introduced by using DeepSeekV3-generated reference answers, particularly in metrics that rely on semantic similarity (e.g., BERTscore, BLEURT). Such bias underscores the need for more neutral or human-authored reference sets to ensure fair evaluation.

Our metric-level analysis further highlights the limitations of traditional lexical metrics such as BLEU, which failed to capture meaningful correctness due to the low incidence of n-gram overlap. Semantic metrics like BLEURT performed better, yet even they showed only modest correlation with human judgment, suggesting the need for domain-adapted evaluation frameworks capable of handling symbolic reasoning, mathematical correctness, and terminology fidelity.

Additionally, our qualitative inspection revealed distinct behavioral differences between models. DeepSeekV3 occasionally returned correct answers without full context, potentially reflecting memorization, while DeepSeekR1 more frequently recognized its limitations, refusing to answer when information was insufficient. While such behavior is encouraging from a reliability standpoint, it also emphasizes the inconsistent handling of uncertainty across model architectures.

Taken together, these observations indicate that current large language models (LLMs) lack the domain robustness, contextual fidelity, and reasoning depth required for high-stakes applications in cryptography. They are prone to overgeneralization, factual errors, and reasoning failures when faced with technical specificity or incomplete information.

6.2 Limitations of CryptoQA

In this section, we go into the limitations of our work:

- **Questions and answers:** One of the key limitations of our evaluation framework lies in the brevity and simplicity of the reference answers. The reference responses are relatively short and often lack the depth necessary to capture the full range of information required for complex, multi-step reasoning tasks. Moreover, for many open-ended questions in cryptography, multiple valid answers may exist—each differing in terminology, structure, or emphasis, yet still being semantically and technically correct. In such cases, relying on a single reference answer for evaluation introduces a strong bias and limits the effectiveness of automatic metrics, particularly those that depend on surface-level or semantic similarity (e.g., BLEU, BERTscore). To robustly evaluate such questions, multiple reference answers are needed to better capture the range of acceptable responses and reduce the risk of penalizing legitimate variations in expression. In addition, despite careful filtering and manual inspection, a significant number of context-dependent questions remain in the dataset. These questions cannot be accurately answered without access to the original source material from which they were derived, as they often include references to specific sections, terminology, or examples that are not self-contained in the question. This residual presence of contextual dependencies further reduces the reliability of the evaluation and highlights the need for more rigorous context-awareness in both dataset construction and model prompting. Collectively, these limitations point to the need for enhanced dataset design, including the development of multi-reference, context-resolved benchmarks that are capable of capturing the complexity and variability inherent in cryptographic knowledge tasks.
- **Evaluation:** The overall quality and reliability of our benchmark are strongly contingent upon the choice of evaluation metrics. In the context of free-text, open-domain question answering (QA), identifying metrics that consistently align with human judgment remains an open and challenging research problem [50]. Consequently, the interpretation of our results must be viewed through the lens of the limitations inherent in current evaluation methodologies. In our study, we employed a combination of lexical matching metrics (such as BLEU and ROUGE) alongside neural-based semantic evaluation metrics (including BERTscore and BLEURT) [50]. While lexical matching methods have historically been popular, they suffer from several critical shortcomings: they fail to accommodate synonyms, paraphrasing, and variations in word order, which are commonplace in free-text answers [50]. This limitation results in an underestimation of answer correctness when the candidate and reference use different but semantically equivalent expressions. Furthermore, neural evaluation metrics attempt to address some of these issues by measuring the semantic similarity between candidate and reference answers through contextual embeddings. However, these approaches also have inherent drawbacks. Primarily, they do not explicitly evaluate the factual correctness of an answer, but rather its proximity in embedding space to the reference. This reliance on embeddings can sometimes misrepresent correctness, especially in technical domains where subtle distinctions matter. Furthermore, neural metrics often require the setting of an arbitrary threshold to classify answers as correct or incorrect, complicating their interpretability and practical deployment [50].

6.2.1 Future Work

To bridge this gap, further research is needed in several directions: Datasets, models, and evaluation.

For datasets:

- To better accommodate the diversity of valid answers in open-ended cryptographic questions, future datasets should include multiple reference answers for each question. This would help mitigate evaluation biases introduced by single-reference comparisons and improve the robustness of automatic metrics.
- To encourage deeper comprehension, benchmarks might incorporate requirements for step-by-step reasoning or explanatory answers rather than brief factual statements. This would enable evaluation of models' ability to perform complex logical deductions and provide transparency in their reasoning.
- To create benchmarks covering the application of several encryptions schemes.

For models:

- Context-aware architectures, such as retrieval-augmented generation or memory-augmented models, could help ground answers in the original source material.
- Domain-specific fine-tuning on curated cryptographic corpora may improve factual accuracy and terminology use.
- Transparency and uncertainty modeling, including calibrated refusal to answer when uncertain, should be further explored to enhance trustworthiness.

For evaluation:

- Human-in-the-loop evaluation and improved metrics tailored for technical reasoning are essential to better assess and guide model behavior.
- It is important to note that our evaluation did not include LLM-as-a-judge approaches, such as G-eval[29], which leverage the reasoning capabilities of large language models to assess answer quality. Although such methods have shown improved alignment with human judgment, they present significant challenges including high computational cost, dependence on careful prompt engineering, and potential variability in scoring [50]. Moreover, obtaining human judgment at scale for a dataset of this magnitude is practically infeasible due to the substantial time, expertise, and resources required. Human evaluation remains the gold standard for assessing answer correctness and quality, but the scale of modern LLM benchmarks often prohibits comprehensive manual assessment. To meaningfully interpret LLM performance, establishing a human baseline is crucial. However, the sheer size of datasets like ours renders this endeavor almost unattainable in practice, thereby necessitating continued research into robust, scalable, and domain-sensitive evaluation methods that better approximate human judgment.

These directions collectively aim to address current shortcomings in dataset, models, and evaluation methodology, thereby fostering more reliable, interpretable, and practically useful benchmarks for LLMs in cryptographic question answering.

7 Conclusion

In this work, we introduce a novel cryptography question answering dataset consisting of over one million samples, designed to facilitate both the training and comprehensive evaluation of large language models (LLMs) within the specialized domain of cryptography. This dataset represents one of the largest and most diverse corpora available for cryptographic QA, encompassing a wide array of question types, difficulty levels, and topical coverage.

This dataset was curated by collecting high-quality sources and leveraging DeepSeekV3 to generate question answer (QA)-pairs. The automatic generation of QA-pairs allowed us to generate significantly more questions compared to human-labeled data.

By providing a substantial volume of high-quality data, our dataset enables LLMs to learn intricate cryptographic concepts, terminologies, and reasoning patterns that are essential for accurate and reliable question answering. Furthermore, its scale and diversity support rigorous benchmarking across multiple evaluation metrics and scenarios, promoting the development and assessment of models that can better understand and generate domain-specific knowledge.

In conclusion, we found that while LLMs exhibit baseline competence in cryptographic QA, they are not yet ready for reliable or autonomous use in cryptography education, research, or practice. Continued advancements in architecture, training paradigms, and evaluation methodologies will be crucial to meet the precision and reliability standards this domain demands.

Acknowledgement

The authors acknowledge support by the state of Baden-Württemberg through bwHPC.

Bibliography

[1] C. C. Aggarwal. *Neural Networks and Deep Learning. A Textbook*. 2nd ed. Springer Cham, June 29, 2023, pp. XXIV, 529. ISBN: 978-3-031-29642-0. DOI: 10.1007/978-3-031-29642-0. URL: https://link.springer.com/book/10.1007/978-3-031-29642-0 (cit. on p. 21).

[2] A. Alismail, C. Lanquillon. "A Survey of LLM-Based Methods for Synthetic Data Generation and the Rise of Agentic Workflows". In: *International Conference on Human-Computer Interaction*. Springer. 2025, pp. 119–135 (cit. on p. 38).

[3] A. Amini, S. Gabriel, P. Lin, R. Koncel-Kedziorski, Y. Choi, H. Hajishirzi. "Mathqa: Towards interpretable math word problem solving with operation-based formalisms". In: *arXiv preprint arXiv:1905.13319* (2019) (cit. on p. 32).

[4] A. Baevski, S. Edunov, Y. Liu, L. Zettlemoyer, M. Auli. "Cloze-driven pretraining of self-attention networks". In: *arXiv preprint arXiv:1903.07785* (2019) (cit. on p. 23).

[5] D. Bahdanau, K. Cho, Y. Bengio. "NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE". In: *arXiv preprint arXiv:1409.0473* (2014). URL: https://wtf-deeplearning.github.io/machine-translation/1409.0473.pdf (visited on 06/17/2025) (cit. on p. 20).

[6] M. Barman. *CryptoLLM - Large Language Models forCryptography Assistance*. Tech. rep. University of Stuttgart, May 17, 2024 (cit. on pp. 31, 39).

[7] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901 (cit. on pp. 19, 21, 22).

[8] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. "Training verifiers to solve math word problems". In: *arXiv preprint arXiv:2110.14168* (2021) (cit. on pp. 17, 27, 32).

[9] DeepSeek-AI. *DeepSeek-V3 Technical Report*. 2024. arXiv: 2412.19437 [cs.CL]. URL: https://arxiv.org/abs/2412.19437 (cit. on pp. 19, 21, 22, 38).

[10] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 2019, pp. 4171–4186 (cit. on pp. 21, 22, 24).

[11] L. Fahrmeir, C. Heumann, R. Künstler, I. Pigeot, G. Tutz. *Statistik: Der Weg zur Datenanalyse*. Springer-Verlag, 2016 (cit. on p. 51).

[12] W. Fan, Y. Ding, L. Ning, S. Wang, H. Li, D. Yin, T.-S. Chua, Q. Li. "A survey on rag meeting llms: Towards retrieval-augmented large language models". In: *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*. 2024, pp. 6491–6501 (cit. on p. 27).

[13] G. Feng, K. Yang, Y. Gu, X. Ai, S. Luo, J. Sun, D. He, Z. Li, L. Wang. “How numerical precision affects mathematical reasoning capabilities of llms”. In: *arXiv preprint arXiv:2410.13857* (2024) (cit. on p. 27).

[14] M. A. Ferrag, F. Alwahedi, A. Battah, B. Cherif, A. Mechri, N. Tihanyi, T. Bisztray, M. Debbah. *Generative AI in Cybersecurity: A Comprehensive Review of LLM Applications and Vulnerabilities*. 2025. arXiv: 2405.12750 [cs.CR]. URL: https://arxiv.org/abs/2405.12750 (cit. on p. 17).

[15] A. Gambardella, Y. Iwasawa, Y. Matsuo. “Language models do hard arithmetic tasks easily and hardly do easy arithmetic tasks”. In: *arXiv preprint arXiv:2406.02356* (2024) (cit. on p. 27).

[16] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, N. A. Smith. “Don’t stop pretraining: Adapt language models to domains and tasks”. In: *arXiv preprint arXiv:2004.10964* (2020) (cit. on p. 25).

[17] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, J. Steinhardt. “Measuring massive multitask language understanding”. In: *arXiv preprint arXiv:2009.03300* (2020) (cit. on p. 34).

[18] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, J. Steinhardt. *Measuring Mathematical Problem Solving With the MATH Dataset*. 2021. arXiv: 2103.03874 [cs.LG]. URL: https://arxiv.org/abs/2103.03874 (cit. on p. 34).

[19] A. Holtzman, J. Buys, L. Du, M. Forbes, Y. Choi. “The curious case of neural text degeneration”. In: *arXiv preprint arXiv:1904.09751* (2019) (cit. on p. 23).

[20] J. Jiang, F. Wang, J. Shen, S. Kim, S. Kim. *A Survey on Large Language Models for Code Generation*. 2024. arXiv: 2406.00515 [cs.CL]. URL: https://arxiv.org/abs/2406.00515 (cit. on p. 17).

[21] K. S. Kalyan, A. Rajasekharan, S. Sangeetha. “Ammus: A survey of transformer-based pretrained models in natural language processing”. In: *arXiv preprint arXiv:2108.05542* (2021) (cit. on pp. 17, 23–25, 27).

[22] U. Kamath, K. Keenan, G. Somers, S. Sorenson. *Large Language Models: A Deep Dive. Bridging Theory and Practice*. 1st ed. Springer Cham, Aug. 20, 2024, pp. XXXIV, 472. ISBN: 978-3-031-65647-7. DOI: https://doi.org/10.1007/978-3-031-65647-7. URL: https://doi.org/10.1007/978-3-031-65647-7 (cit. on pp. 19–21, 23–28).

[23] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa. “Large language models are zero-shot reasoners”. In: *Advances in neural information processing systems* 35 (2022), pp. 22199–22213 (cit. on p. 26).

[24] N. Kushman, Y. Artzi, L. Zettlemoyer, R. Barzilay. “Learning to automatically solve algebra word problems”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2014, pp. 271–281 (cit. on p. 32).

[25] R. Küsters, T. Wilke. *Moderne Kryptographie*. Springer, 2011 (cit. on p. 17).

[26] Y. Li, Q. Pei, M. Sun, H. Lin, C. Ming, X. Gao, J. Wu, C. He, L. Wu. *CipherBank: Exploring the Boundary of LLM Reasoning Capabilities through Cryptography Challenges*. 2025. arXiv: 2504.19093 [cs.CR]. URL: https://arxiv.org/abs/2504.19093 (cit. on pp. 17, 31).

[27] C.-Y. Lin. “Rouge: A package for automatic evaluation of summaries”. In: *Text summarization branches out*. 2004, pp. 74–81 (cit. on pp. 28, 44).

[28] W. Ling, D. Yogatama, C. Dyer, P. Blunsom. “Program induction by rationale generation: Learning to solve and explain algebraic word problems”. In: *arXiv preprint arXiv:1705.04146* (2017) (cit. on p. 32).

[29] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, C. Zhu. “G-eval: NLG evaluation using gpt-4 with better human alignment”. In: *arXiv preprint arXiv:2303.16634* (2023) (cit. on p. 63).

[30] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov. “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692* (2019) (cit. on p. 25).

[31] Z. Liu. “Secqa: A concise question-answering dataset for evaluating large language models in computer security”. In: *arXiv preprint arXiv:2312.15838* (2023) (cit. on p. 33).

[32] U. Maskey, C. Zhu, U. Naseem. “Benchmarking large language models for cryptanalysis and mismatched-generalization”. In: *arXiv preprint arXiv:2505.24621* (2025) (cit. on p. 31).

[33] D. Noever. “LARGE LANGUAGE MODELS FOR CIPHERS”. In: *International Journal of Artificial Intelligence & Applications* 14 (May 2023), pp. 1–20. DOI: 10.5121/ijaia.2023.14301 (cit. on p. 31).

[34] T. Okadome. “Essentials of Generative AI”. In: (2025). DOI: 10.1007/978-981-96-0029-8 (cit. on pp. 19–21).

[35] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu. “Bleu: a method for automatic evaluation of machine translation”. In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002, pp. 311–318 (cit. on pp. 28, 44).

[36] S. J. Prince. *Understanding deep learning*. MIT press, 2023 (cit. on p. 22).

[37] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu. “Exploring the limits of transfer learning with a unified text-to-text transformer”. In: *Journal of machine learning research* 21.140 (2020), pp. 1–67 (cit. on pp. 21, 23, 24).

[38] A. Rogers, M. Gardner, I. Augenstein. “Qa dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension”. In: *ACM Computing Surveys* 55.10 (2023), pp. 1–45 (cit. on pp. 31, 42).

[39] S. Roy, D. Roth. “Solving general arithmetic word problems”. In: *arXiv preprint arXiv:1608.01413* (2016) (cit. on p. 32).

[40] S. Roy, T. Vieira, D. Roth. “Reasoning about quantities in natural language”. In: *Transactions of the Association for Computational Linguistics* 3 (2015), pp. 1–13 (cit. on p. 32).

[41] T. Sellam, D. Das, A. P. Parikh. “BLEURT: Learning robust metrics for text generation”. In: *arXiv preprint arXiv:2004.04696* (2020) (cit. on pp. 29, 44, 45).

[42] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, J. Dean. “Outrageously large neural networks: The sparsely-gated mixture-of-experts layer”. In: *arXiv preprint arXiv:1701.06538* (2017) (cit. on p. 22).

[43] A. Srivastava et al. *Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models*. 2023. arXiv: 2206.04615 [cs.CL]. URL: https://arxiv.org/abs/2206.04615 (cit. on p. 33).

[44] A. Srivastava, A. Memon. “Towards Robust Evaluation: A Comprehensive Taxonomy of Datasets and Metrics for Open Domain Question Answering in the Era of Large Language Models”. In: *IEEE Access* (2024) (cit. on p. 44).

[45] J. M. Tomczak. “Deep Generative Modeling”. In: (2024). DOI: 10.1007/978-3-031-64087-2 (cit. on p. 19).

[46] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 20–22).

[47] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, S. Bowman. “Superglue: A stickier benchmark for general-purpose language understanding systems”. In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 34).

[48] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, S. R. Bowman. “GLUE: A multi-task benchmark and analysis platform for natural language understanding”. In: *arXiv preprint arXiv:1804.07461* (2018) (cit. on p. 34).

[49] B. Wang, C. Xu, X. Zhao, L. Ouyang, F. Wu, Z. Zhao, R. Xu, K. Liu, Y. Qu, F. Shang, B. Zhang, L. Wei, Z. Sui, W. Li, B. Shi, Y. Qiao, D. Lin, C. He. *MinerU: An Open-Source Solution for Precise Document Content Extraction*. 2024. arXiv: 2409.18839 [cs.CV]. URL: https://arxiv.org/abs/2409.18839 (cit. on p. 37).

[50] C. Wang, S. Cheng, Q. Guo, Y. Yue, B. Ding, Z. Xu, Y. Wang, X. Hu, Z. Zhang, Y. Zhang. “Evaluating open-qa evaluation”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 77013–77042 (cit. on pp. 45, 62, 63).

[51] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. “Chain-of-thought prompting elicits reasoning in large language models”. In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837 (cit. on pp. 17, 26, 27).

[52] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, C.-J. Hsieh. “Large batch optimization for deep learning: Training bert in 76 minutes”. In: *arXiv preprint arXiv:1904.00962* (2019) (cit. on p. 25).

[53] D. Yu, R. Bao, R. Ning, J. Peng, G. Mai, L. Zhao. *Spatial-RAG: Spatial Retrieval Augmented Generation for Real-World Geospatial Reasoning Questions*. 2025. arXiv: 2502.18470 [cs.IR]. URL: https://arxiv.org/abs/2502.18470 (cit. on p. 26).

[54] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, Y. Artzi. “Bertscore: Evaluating text generation with bert”. In: *arXiv preprint arXiv:1904.09675* (2019) (cit. on pp. 29, 50).

[55] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. “A survey of large language models”. In: *arXiv preprint arXiv:2303.18223* 1.2 (2023) (cit. on pp. 23, 26).

All links were last followed on March 17, 2018.

A Appendix

Link to the filter of the university library

https://rds-stg.ibs-bw.de/opac/RDSIndex/Search?join=AND&bool0[]=AND&lookfor0[]=Cryptography&lookfor0[]=Springer&type0[]=allfields&type0[]=pu&sort=py+asc%2C+title&limit=20&filter[]=mt_facet%3A%22online-ressource%22&filter[]=la_facet%3A%22eng%22&filter[]=-py_facet%3A%220%22&filter[]=-mt_facet%3A%22hs%22&filter[]=-mt_facet%3A%22fest%22

Keywords for filtering

- paper
- Theorems? [1-9]
- future
- author
- funding
- dataset
- Lemma
- study
- conclusion
- hardware
- chapter
- article
- training

B Statistics

		Book	Conference proceedings	Textbook	math	word	Total
Occurrences		14505	190870	6908	40210	172073	212283
bleu_v3	mean	0.0208	0.0093	0.0472	0.0141	0.0107	0.0113
	std	0.0547	0.0354	0.0892	0.0484	0.0384	0.0405
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	50%	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	75%	0.0000	0.0000	0.0746	0.0000	0.0000	0.0000
	max	0.7612	1.0000	0.9036	0.9036	1.0000	1.0000
bleu_r1	mean	0.0142	0.0070	0.0332	0.0101	0.0079	0.0084
	std	0.0423	0.0300	0.0683	0.0388	0.0318	0.0333
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	50%	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	75%	0.0000	0.0000	0.0522	0.0000	0.0000	0.0000
	max	1.0000	1.0000	0.9036	0.9036	1.0000	1.0000
bleurt_v3	mean	0.4770	0.4326	0.5079	0.4140	0.4438	0.4381
	std	0.0965	0.0880	0.0962	0.0817	0.0915	0.0905
	min	0.1009	-0.0188	-0.0435	0.0320	-0.0435	-0.0435
	25%	0.4076	0.3730	0.4372	0.3597	0.3813	0.3763
	50%	0.4715	0.4268	0.5042	0.4061	0.4383	0.4316
	75%	0.5430	0.4884	0.5759	0.4606	0.5029	0.4956
	max	0.8933	0.9799	0.8914	0.8914	0.9799	0.9799
bleurt_r1	mean	0.4692	0.4295	0.5010	0.4164	0.4387	0.4345
	std	0.0894	0.0819	0.0891	0.0764	0.0853	0.0842
	min	-0.0046	-0.0322	-0.0059	-0.0160	-0.0322	-0.0322
	25%	0.4067	0.3743	0.4359	0.3665	0.3807	0.3775
	50%	0.4640	0.4239	0.4963	0.4098	0.4335	0.4284
	75%	0.5295	0.4805	0.5620	0.4601	0.4928	0.4870
	max	0.9926	0.9631	0.9584	0.9584	0.9926	0.9926
Bert_precision_r1	mean	0.5724	0.5274	0.5913	0.5173	0.5361	0.5326
	std	0.0704	0.0673	0.0731	0.0769	0.0672	0.0695
	min	0.2413	0.1884	0.2902	0.1884	0.1926	0.1884
	25%	0.5262	0.4850	0.5440	0.4695	0.4928	0.4883

Continued on next page

		Book	Conference proceedings	Textbook	math	word	Total
	50%	0.5735	0.5272	0.5898	0.5156	0.5352	0.5317
	75%	0.6179	0.5698	0.6389	0.5633	0.5784	0.5761
	max	1.0000	1.0000	0.9897	0.9956	1.0000	1.0000
Bert_recall_r1	mean	0.6673	0.6423	0.6847	0.6156	0.6523	0.6454
	std	0.0851	0.0868	0.0849	0.1025	0.0816	0.0872
	min	0.2199	0.1998	0.2243	0.1998	0.2243	0.1998
	25%	0.6150	0.5878	0.6329	0.5491	0.5993	0.5905
	50%	0.6702	0.6423	0.6890	0.6160	0.6511	0.6456
	75%	0.7245	0.6985	0.7428	0.6826	0.7059	0.7024
	max	1.0000	1.0000	0.9903	0.9956	1.0000	1.0000
Bert_F1_r1	mean	0.6142	0.5771	0.6326	0.5592	0.5866	0.5814
	std	0.0699	0.0684	0.0717	0.0795	0.0664	0.0699
	min	0.2411	0.2422	0.2801	0.2422	0.2411	0.2411
	25%	0.5686	0.5327	0.5852	0.5070	0.5423	0.5358
	50%	0.6146	0.5759	0.6336	0.5572	0.5843	0.5799
	75%	0.6606	0.6207	0.6799	0.6093	0.6292	0.6261
	max	1.0000	1.0000	0.9712	0.9956	1.0000	1.0000
Bert_precision_v3	mean	0.5964	0.5469	0.6151	0.5304	0.5577	0.5525
	std	0.0688	0.0665	0.0742	0.0798	0.0652	0.0690
	min	0.2612	0.1971	0.2580	0.1971	0.2503	0.1971
	25%	0.5531	0.5064	0.5667	0.4800	0.5167	0.5099
	50%	0.5956	0.5469	0.6127	0.5277	0.5560	0.5515
	75%	0.6392	0.5876	0.6615	0.5771	0.5971	0.5943
	max	0.9890	1.0000	0.9897	0.9897	1.0000	1.0000
Bert_recall_v3	mean	0.6719	0.6458	0.6885	0.6131	0.6573	0.6489
	std	0.0911	0.0930	0.0926	0.1142	0.0857	0.0934
	min	0.3037	0.2217	0.2851	0.2420	0.2217	0.2217
	25%	0.6176	0.5887	0.6311	0.5345	0.6025	0.5916
	50%	0.6758	0.6471	0.6942	0.6138	0.6566	0.6503
	75%	0.7344	0.7067	0.7522	0.6910	0.7139	0.7107
	max	0.9890	1.0000	0.9903	0.9854	1.0000	1.0000
Bert_F1_v3	mean	0.6299	0.5901	0.6478	0.5654	0.6016	0.5947
	std	0.0722	0.0711	0.0760	0.0863	0.0673	0.0727
	min	0.2829	0.2449	0.3412	0.2449	0.2484	0.2449
	25%	0.5849	0.5453	0.5986	0.5065	0.5573	0.5487
	50%	0.6315	0.5903	0.6490	0.5634	0.5999	0.5945
	75%	0.6775	0.6357	0.6969	0.6205	0.6447	0.6414
	max	0.9890	1.0000	0.9712	0.9854	1.0000	1.0000
rouge1_v3	mean	0.2617	0.1948	0.3323	0.2114	0.2021	0.2039
	std	0.1269	0.1093	0.1476	0.1375	0.1099	0.1157

Continued on next page

		Book	Conference proceedings	Textbook	math	word	Total
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.1714	0.1176	0.2273	0.1111	0.1250	0.1224
	50%	0.2462	0.1795	0.3168	0.1892	0.1867	0.1875
	75%	0.3373	0.2553	0.4225	0.2857	0.2632	0.2667
	max	1.0000	1.0000	1.0000	0.9565	1.0000	1.0000
rouge2_v3	mean	0.0908	0.0594	0.1473	0.0803	0.0607	0.0644
	std	0.0992	0.0784	0.1347	0.1017	0.0791	0.0842
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.0244	0.0000	0.0482	0.0000	0.0000	0.0000
	50%	0.0615	0.0333	0.1132	0.0476	0.0339	0.0351
	75%	0.1316	0.0870	0.2105	0.1149	0.0896	0.0938
	max	1.0000	1.0000	1.0000	0.9524	1.0000	1.0000
rougeL_v3	mean	0.2026	0.1558	0.2645	0.1780	0.1589	0.1625
	std	0.1121	0.0954	0.1380	0.1218	0.0948	0.1008
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.1231	0.0909	0.1644	0.0923	0.0937	0.0930
	50%	0.1818	0.1379	0.2414	0.1538	0.1404	0.1429
	75%	0.2581	0.2000	0.3385	0.2381	0.2041	0.2105
	max	1.0000	1.0000	1.0000	0.9565	1.0000	1.0000
rougeLsum_v3	mean	0.2026	0.1558	0.2641	0.1776	0.1590	0.1625
	std	0.1116	0.0952	0.1368	0.1207	0.0948	0.1005
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.1231	0.0909	0.1644	0.0923	0.0938	0.0930
	50%	0.1818	0.1379	0.2414	0.1538	0.1408	0.1429
	75%	0.2588	0.2022	0.3385	0.2381	0.2041	0.2105
	max	1.0000	1.0000	1.0000	0.9565	1.0000	1.0000
rouge1_r1	mean	0.2463	0.1848	0.3143	0.2031	0.1909	0.1932
	std	0.1221	0.1055	0.1388	0.1280	0.1069	0.1113
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.1573	0.1096	0.2128	0.1111	0.1154	0.1143
	50%	0.2321	0.1707	0.3043	0.1842	0.1758	0.1772
	75%	0.3188	0.2439	0.4000	0.2750	0.2500	0.2545
	max	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
rouge2_r1	mean	0.0801	0.0533	0.1326	0.0729	0.0542	0.0577
	std	0.0890	0.0716	0.1200	0.0909	0.0723	0.0765
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.0190	0.0000	0.0424	0.0000	0.0000	0.0000
	50%	0.0548	0.0308	0.1039	0.0444	0.0308	0.0323
	75%	0.1163	0.0784	0.1895	0.1053	0.0800	0.0842
	max	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Continued on next page

		Book	Conference proceedings	Textbook	math	word	Total
rougeL_r1	mean	0.1893	0.1469	0.2476	0.1686	0.1495	0.1531
	std	0.1052	0.0905	0.1270	0.1114	0.0907	0.0952
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.1149	0.0845	0.1558	0.0909	0.0870	0.0870
	50%	0.1702	0.1304	0.2268	0.1481	0.1333	0.1351
	75%	0.2432	0.1905	0.3188	0.2222	0.1935	0.2000
	max	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
rougeLsum_r1	mean	0.1919	0.1486	0.2502	0.1691	0.1515	0.1548
	std	0.1046	0.0909	0.1256	0.1108	0.0912	0.0955
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.1176	0.0857	0.1579	0.0909	0.0882	0.0889
	50%	0.1739	0.1319	0.2316	0.1493	0.1348	0.1370
	75%	0.2462	0.1928	0.3226	0.2258	0.1961	0.2000
	max	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Table B.1: Scores of the models aggregated by book type and category

A: Application, AC Attacks and Cryptanalysis, CP : Cryptographic Protocols, F: Foundations, PKC: Public-Key Cryptography, SKC: Secret-Key Cryptography, O: other, T: over all topics

		A	AC	CP	F	I	PKC	SKC	O	T
bleu_v3	mean	0.0093	0.0072	0.0100	0.0198	0.0077	0.0112	0.0142	0.0146	0.0113
	std	0.0374	0.0304	0.0343	0.0549	0.0329	0.0380	0.0465	0.0470	0.0405
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	50%	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	75%	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	max	0.9254	0.6432	0.5190	0.9036	1.0000	0.4884	0.7506	0.5236	1.0000
bleu_r1	mean	0.0073	0.0054	0.0074	0.0140	0.0060	0.0084	0.0100	0.0111	0.0084
	std	0.0332	0.0250	0.0284	0.0425	0.0288	0.0311	0.0351	0.0414	0.0333
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	50%	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	75%	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	max	0.8932	0.8409	1.0000	0.9036	1.0000	0.4453	0.6340	1.0000	1.0000
bleurt_v3	mean	0.4500	0.4296	0.4386	0.4543	0.4245	0.4347	0.4349	0.4555	0.4381
	std	0.0996	0.0878	0.0866	0.0907	0.0863	0.0881	0.0866	0.0972	0.0905
	min	0.0501	0.0540	0.0106	-0.0435	-0.0099	0.0920	0.0921	0.0995	-0.0435
	25%	0.3823	0.3701	0.3784	0.3904	0.3667	0.3738	0.3751	0.3897	0.3763
	50%	0.4478	0.4224	0.4321	0.4469	0.4193	0.4249	0.4247	0.4509	0.4316
	75%	0.5172	0.4842	0.4933	0.5132	0.4788	0.4895	0.4878	0.5201	0.4956
	max	0.8784	0.8298	0.8350	0.8919	0.9799	0.7849	0.9486	0.8933	0.9799
bleurt_r1	mean	0.4439	0.4266	0.4348	0.4500	0.4219	0.4321	0.4342	0.4506	0.4345
	std	0.0919	0.0809	0.0810	0.0846	0.0806	0.0813	0.0807	0.0902	0.0842
	min	0.0039	0.0203	0.0437	0.0017	-0.0322	-0.0059	0.1192	-0.0160	-0.0322
	25%	0.3832	0.3716	0.3791	0.3910	0.3685	0.3771	0.3787	0.3900	0.3775
	50%	0.4409	0.4193	0.4290	0.4433	0.4170	0.4243	0.4259	0.4471	0.4284
	75%	0.5044	0.4758	0.4851	0.5034	0.4712	0.4814	0.4813	0.5076	0.4870
	max	0.8832	0.7932	0.9631	0.9345	0.9354	0.7977	0.9584	0.9926	0.9926
Bert_precision_r1	mean	0.5373	0.5231	0.5355	0.5406	0.5265	0.5336	0.5354	0.5366	0.5326
	std	0.0749	0.0646	0.0627	0.0697	0.0704	0.0648	0.0663	0.0767	0.0695
	min	0.2239	0.2215	0.1884	0.2337	0.2026	0.2474	0.2581	0.1926	0.1884
	25%	0.4883	0.4828	0.4948	0.4943	0.4831	0.4916	0.4926	0.4887	0.4883
	50%	0.5369	0.5221	0.5346	0.5384	0.5270	0.5319	0.5349	0.5374	0.5317
	75%	0.5847	0.5624	0.5763	0.5844	0.5707	0.5754	0.5771	0.5840	0.5761
	max	0.9923	0.9526	1.0000	0.9897	1.0000	0.9529	0.8555	1.0000	1.0000
Bert_recall_r1	mean	0.6563	0.6333	0.6422	0.6550	0.6395	0.6412	0.6404	0.6605	0.6454
	std	0.0846	0.0822	0.0834	0.0882	0.0887	0.0873	0.0900	0.0894	0.0872
	min	0.2243	0.2848	0.2790	0.1998	0.2073	0.2320	0.3254	0.2478	0.1998
	25%	0.6026	0.5817	0.5904	0.5991	0.5840	0.5881	0.5849	0.6049	0.5905
	50%	0.6567	0.6318	0.6432	0.6554	0.6397	0.6418	0.6424	0.6609	0.6456
	75%	0.7115	0.6863	0.6970	0.7142	0.6966	0.6991	0.6994	0.7193	0.7024
	max	0.9923	0.9645	1.0000	0.9936	1.0000	0.9474	0.9435	1.0000	1.0000

Continued on next page

		A	AC	CP	F	I	PKC	SKC	O	T
Bert_F1_r1	mean	0.5888	0.5711	0.5822	0.5902	0.5751	0.5805	0.5810	0.5898	0.5814
	std	0.0724	0.0654	0.0651	0.0706	0.0703	0.0679	0.0692	0.0750	0.0699
	min	0.2411	0.2761	0.2643	0.2557	0.2422	0.2801	0.2978	0.2482	0.2411
	25%	0.5413	0.5292	0.5390	0.5430	0.5301	0.5354	0.5348	0.5430	0.5358
	50%	0.5875	0.5693	0.5808	0.5879	0.5743	0.5786	0.5805	0.5897	0.5799
	75%	0.6353	0.6116	0.6250	0.6362	0.6198	0.6253	0.6261	0.6381	0.6261
	max	0.9923	0.9343	1.0000	0.9836	1.0000	0.9378	0.8944	1.0000	1.0000
Bert_precision_v3	mean	0.5659	0.5406	0.5517	0.5622	0.5448	0.5481	0.5503	0.5621	0.5525
	std	0.0725	0.0642	0.0620	0.0702	0.0688	0.0664	0.0682	0.0763	0.0690
	min	0.2663	0.2476	0.2666	0.2574	0.1971	0.2559	0.2652	0.2532	0.1971
	25%	0.5220	0.5016	0.5121	0.5167	0.5042	0.5041	0.5077	0.5161	0.5099
	50%	0.5660	0.5397	0.5506	0.5588	0.5460	0.5458	0.5486	0.5632	0.5515
	75%	0.6108	0.5785	0.5906	0.6039	0.5865	0.5894	0.5910	0.6110	0.5943
	max	0.9979	0.9526	0.9006	0.9897	1.0000	0.8445	0.9890	0.9081	1.0000
Bert_recall_v3	mean	0.6662	0.6342	0.6422	0.6593	0.6434	0.6382	0.6377	0.6674	0.6489
	std	0.0874	0.0886	0.0892	0.0956	0.0942	0.0957	0.1001	0.0945	0.0934
	min	0.2394	0.2421	0.2624	0.2732	0.2217	0.3083	0.2708	0.2792	0.2217
	25%	0.6109	0.5803	0.5891	0.6000	0.5854	0.5811	0.5761	0.6087	0.5916
	50%	0.6676	0.6341	0.6447	0.6611	0.6451	0.6397	0.6408	0.6702	0.6503
	75%	0.7249	0.6909	0.7003	0.7239	0.7045	0.7026	0.7028	0.7335	0.7107
	max	0.9979	0.9763	0.9567	0.9921	1.0000	0.9293	0.9890	0.9814	1.0000
Bert_F1_v3	mean	0.6100	0.5818	0.5917	0.6048	0.5875	0.5876	0.5885	0.6080	0.5947
	std	0.0724	0.0683	0.0678	0.0746	0.0718	0.0729	0.0754	0.0772	0.0727
	min	0.2522	0.2544	0.2677	0.3256	0.2449	0.3243	0.2679	0.2796	0.2449
	25%	0.5638	0.5392	0.5490	0.5564	0.5426	0.5406	0.5407	0.5588	0.5487
	50%	0.6107	0.5812	0.5912	0.6037	0.5886	0.5864	0.5899	0.6096	0.5945
	75%	0.6576	0.6247	0.6358	0.6530	0.6332	0.6366	0.6352	0.6599	0.6414
	max	0.9979	0.9343	0.9278	0.9871	1.0000	0.8571	0.9890	0.9433	1.0000
rouge1_v3	mean	0.1954	0.1802	0.2055	0.2397	0.1865	0.2104	0.2235	0.2171	0.2039
	std	0.1110	0.1016	0.1071	0.1294	0.1069	0.1158	0.1242	0.1270	0.1157
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.1176	0.1071	0.1290	0.1463	0.1111	0.1270	0.1348	0.1290	0.1224
	50%	0.1818	0.1667	0.1905	0.2222	0.1739	0.1939	0.2069	0.1967	0.1875
	75%	0.2571	0.2368	0.2667	0.3117	0.2456	0.2737	0.2903	0.2857	0.2667
	max	0.9787	0.8837	0.9032	1.0000	1.0000	0.7931	1.0000	1.0000	1.0000
rouge2_v3	mean	0.0561	0.0501	0.0618	0.0899	0.0537	0.0688	0.0775	0.0749	0.0644
	std	0.0772	0.0688	0.0765	0.1025	0.0738	0.0844	0.0950	0.0969	0.0842
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	50%	0.0323	0.0294	0.0351	0.0597	0.0317	0.0400	0.0494	0.0408	0.0351
	75%	0.0833	0.0727	0.0909	0.1311	0.0784	0.1000	0.1111	0.1071	0.0938
	max	0.9286	0.7273	0.8125	1.0000	1.0000	0.6116	0.9375	1.0000	1.0000
rougeL_v3	mean	0.1539	0.1422	0.1611	0.1937	0.1492	0.1657	0.1799	0.1740	0.1625
	std	0.0947	0.0860	0.0920	0.1157	0.0929	0.0989	0.1098	0.1122	0.1008
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Continued on next page

		A	AC	CP	F	I	PKC	SKC	O	T
	25%	0.0889	0.0833	0.0964	0.1111	0.0870	0.0954	0.1053	0.0968	0.0930
	50%	0.1356	0.1250	0.1446	0.1714	0.1333	0.1471	0.1584	0.1509	0.1429
	75%	0.2000	0.1842	0.2062	0.2500	0.1935	0.2133	0.2316	0.2258	0.2105
	max	0.9333	0.8571	0.8889	1.0000	1.0000	0.7222	0.9032	1.0000	1.0000
rougeLsum_v3	mean	0.1540	0.1424	0.1610	0.1936	0.1492	0.1658	0.1798	0.1741	0.1625
	std	0.0947	0.0861	0.0918	0.1151	0.0927	0.0989	0.1092	0.1120	0.1005
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.0889	0.0833	0.0964	0.1111	0.0870	0.0960	0.1053	0.0968	0.0930
	50%	0.1356	0.1263	0.1446	0.1714	0.1333	0.1471	0.1584	0.1509	0.1429
	75%	0.2000	0.1842	0.2069	0.2500	0.1935	0.2133	0.2316	0.2258	0.2105
	max	0.9333	0.8571	0.8889	1.0000	1.0000	0.7000	0.9032	1.0000	1.0000
rouge1_r1	mean	0.1805	0.1713	0.1974	0.2268	0.1773	0.2045	0.2145	0.2024	0.1932
	std	0.1081	0.0978	0.1037	0.1235	0.1032	0.1105	0.1169	0.1238	0.1113
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.1034	0.1013	0.1235	0.1379	0.1053	0.1263	0.1304	0.1154	0.1143
	50%	0.1644	0.1573	0.1842	0.2090	0.1639	0.1875	0.2000	0.1852	0.1772
	75%	0.2410	0.2258	0.2571	0.2979	0.2353	0.2667	0.2812	0.2667	0.2545
	max	0.9091	0.8235	1.0000	1.0000	1.0000	0.7317	1.0000	1.0000	1.0000
rouge2_r1	mean	0.0488	0.0454	0.0564	0.0801	0.0482	0.0639	0.0702	0.0665	0.0577
	std	0.0703	0.0624	0.0701	0.0924	0.0677	0.0774	0.0841	0.0897	0.0765
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	50%	0.0282	0.0274	0.0328	0.0533	0.0290	0.0385	0.0450	0.0348	0.0323
	75%	0.0714	0.0659	0.0842	0.1165	0.0702	0.0923	0.1017	0.0952	0.0842
	max	0.9000	0.8000	1.0000	1.0000	1.0000	0.6667	1.0000	1.0000	1.0000
rougeL_r1	mean	0.1425	0.1342	0.1538	0.1816	0.1409	0.1599	0.1703	0.1620	0.1531
	std	0.0917	0.0811	0.0872	0.1081	0.0883	0.0928	0.1002	0.1082	0.0952
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.0800	0.0789	0.0930	0.1053	0.0811	0.0952	0.1010	0.0882	0.0870
	50%	0.1250	0.1194	0.1379	0.1600	0.1250	0.1429	0.1522	0.1404	0.1351
	75%	0.1860	0.1728	0.1978	0.2366	0.1837	0.2059	0.2198	0.2105	0.2000
	max	0.9091	0.8235	1.0000	1.0000	1.0000	0.7317	1.0000	1.0000	1.0000
rougeLsum_r1	mean	0.1454	0.1358	0.1555	0.1835	0.1422	0.1616	0.1713	0.1648	0.1548
	std	0.0924	0.0817	0.0877	0.1078	0.0885	0.0936	0.1001	0.1089	0.0955
	min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	25%	0.0833	0.0800	0.0938	0.1067	0.0822	0.0952	0.1017	0.0896	0.0889
	50%	0.1290	0.1212	0.1404	0.1628	0.1270	0.1449	0.1538	0.1429	0.1370
	75%	0.1905	0.1754	0.2000	0.2394	0.1847	0.2083	0.2222	0.2154	0.2000
	max	0.9091	0.8235	1.0000	1.0000	1.0000	0.7317	1.0000	1.0000	1.0000

Table B.2: Scores of the models over the different topics

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature