POLITECNICO DI TORINO

Master's Degree in Energy and Nuclear Engineering

Master's Degree Thesis

Nuclear Security: A Natural Language Processing Generative Approach

Supervisors

Prof. Raffaella TESTONI

Dr. Ahmed NAGY

Dr. Johan CAMPS

Candidate

Gabriele IOB

March 2024

Abstract

This thesis aims at investigating and evaluating the use of natural language generative models, specifically in the framework of generating training scenarios for personnel working in critical infrastructures. Safety and security of an infrastructure containing radioactive material should be addressed with emergency planning, which requires training scenarios for personnel involved. Such scenarios are traditionally developed by human experts, although this is a process that is subject to several drawbacks. First of all, it requires a considerable amount of time for producing a qualitative output. Second, it must deal with multiple repetitive steps, thus leading to critical bottlenecks. Third, a final training scenario can at first glance seem reproducible and on-point, but when it comes to put it in practice, its plausibility might be insufficient. These aspects can ultimately lead to considerable decrease in training quality, with severe consequences on overall safety and security. With this in mind, the possibility of using generative methods in such pipeline has been explored in this work. Generative methods employ the use of Large Language Models (LLMs) to generate long and meaningful text from a simple user prompt. These models leverage the power of Machine Learning (ML) techniques to infer commonly occurring patterns in large training datasets.

The use of generative models in scenario development has the potential of streamlining the most tedious steps, thus improving quality and reliability for the final result. Their use can be investigated with two methodologies in mind, where a fully automated or a semi automated framework are both available solutions. In order to assess the quality of the scenario, some key evaluation criteria were selected and defined. After experimenting with multiple implementations and exploring existing literature, a hierarchical framework using a Generative Pre-trained Transformer (GPT) model was developed, with the aim of generating meaningful, complete and usable scenarios. Multiple scenarios were extracted from several open source examples found on internet archives, in particular from drills designed by international agencies. Human experts were then asked to provide a score for each selected evaluation criterion.

What has been observed after the scenario generation is that using a simple prompt, with the bare model, lead to lack of important information, such as a detailed timeline of the exercise. After providing a general structure, scenarios automatically generated presented more adherence to scenarios designed by humans. When Chain of Thought was used as a prompting strategy, outputs presented more detail and adherence to the selected structure.

As a conclusion, the GPT model was able to generate meaningful scenarios, allowing for flexibility in its implementation. When adopting the hierarchical architecture, explicitly describing the context and limiting the working window helped the model to perform better according to the evaluation criteria.

Acknowledgements

First of all, I would like to thank the ENEN2plus mobility project for funding this master thesis. A sincere thank you goes to my mentors Ahmed and Raffaella, for helping me and giving me precious insights, and to all the people of the CMD group.
A special thank you goes to all my family, Papi, Mamma, Fede e Giulia, of course all the cusins, zias e zios, for supporting me, believing in me, trusting me and helping me in basically everything, even when my mood was not the brightest. You had the patience and kindness of being always at my side, which is something that I never took for granted and that I will always carry with me. Thank you!
A warm thank you goes to my second italian family in Mol: Fede, Fede, Fede, Gabri, Matteo, Luca and Luca. You hosted me and welcomed me with a heartwarming kindness, making me feel at home, sharing with me runs, calisthenics and climbing sessions, discussions about the future, lunches at SCK, parties, dinners, yoga practices and bakery activities. Thank you all!!
A big thank you goes to all the beautiful people sharing the pleasures of living in Boeretang. You had the power of making this experience special and unforgettable, crafting joyful memories from that godforsaken land. Thank you guyss!
Of course, this experience would not have been this special without the presence, laughs, dinners and educational trips with The Last Italians, my acquired brothers and sisters: Flavia, Panfilia, Loukas and Luca. The meals together, the Morning episodes, the always-present weekend trips, the groceries, the evening grumbles about a seemingly unreachable thesis.. I could go on but I would get too emotional, so I finish with a simple but immense thank you!
A special thanks goes to Luca, my comrade in this convoluted adventure. We have been the support of each other, and I am still convinced that without you I would not have been able to reach the end, so thank you bro! (P.S. remember to drink tea and stay hydrated).
While these 6 months have been full of new people and experiences, I cannot forget the guys with whom I shared my 2 years of this Master degree, the one and onlies Compagni di merende. We shared a lot of this polytechnic land of misery, we've been through tough times and we managed to survive (more or less), but we had

some great times together, which I sincerely hope will never reach an end, Thank you guys! The guys from LISTA FLY for the parties, drinks and good times; Ginny e Betta (and the others of course) for the climbing sessions; Paolo, Luca, Andrea, Fra (honorable mention), London, Cristian (e Pablo) for sharing that incredible cave of trolls and mystic creatures in Via Cristoforo Colombo 44, Boggio will miss us! Thank you all!!
Thank you Ingegneristi Anonimi, you made engineering an enjoyable activity, which I still think it's something miraculous! Thank you Anna, Andò, Luna, Foschi, Silver, Vale, Buso, Nigro e Crainz for the fun together, the parties and the quality time! Thank you Fil, Buso, Vriz, Cime, Ale, Sbord for the nerd sessions! Thank you Didier, Crainz, Cec e Vernì for the jams (even though they now belong to the past sadly)! Thank you Ari, Mery, Pili, Giona, Campa, Vernì, Emi e Carol for keeping up with the dinners without letting this group fade into the past! Thank you Grassi, Nich, Crainz, Fabio e Ari (e Cime quando vuole) for the beers and keeping up with the hard climbing regime! Thanks to all the guys at Chiodo Fisso for being such friendly and welcoming people! Thanks to Gozz, Paglia e Pelle for the philosophical debates!

Table of Contents

List of Tables

List of Figures

Acronyms

AI Artificial Intelligence.

CBRN Chemical, Biological, Radiological and Nuclear.

CNN Convolutional Neural Network.

CoT Chain-of-Thought.

DL Deep Learning.

FNN Feedforward Neural Network.

LLM Large Language Model.

LSTM Long-Short Term Memory.

ML Machine Learning.

NLG Natural Language Generation.

NLP Natural Language Processing.

NN Neural Network.

RAG Retrieval Augmented Generation.

RLHF Reinforcement Learning from Human Feedback.

RNN Recurrent Neural Network.

RQ Research Question.

ToT Tree-of-Thoughts.

Chapter 1

Introduction

Radioactive materials are an extremely useful resource, used principally for engineering, research and medical aims. Their manipulation must be performed by experts and specifically trained operators, who know the potential hazards that they might encounter. These materials need to be protected from external hazardous events and situations, and that is the goal of security training. Such threats involve theft, external infiltrations or counterfeit material introduction, which represent very dangerous scenarios, either for the operators and for the population. For these reasons, training is essential to educate safety personnel to cope with multiple scenarios, making it less likely that a dangerous situation evolves, or at least mitigating the potential consequences of an already initiated emergency. Training and exercises are one of the principal means for demonstrating the effectiveness of an emergency program.
Such exercises need to be developed by experts in Chemical, Biological, Radiological and Nuclear (CBRN) training, but this has the potential to make the whole procedure tedious to implement inside facilities and critical infrastructures, which in itself constitutes a problem for the safety of the site. With this in mind, a reliable way to automate the development of these scenarios would be of great importance and convenience, while at the same time focusing on the quality of generated scenarios, which should be ideally indistinguishable from expert-based ones.

1.1 Problem statement

CBRN refers to threats "that could harm the environment that could harm the society through their accidental or deliberate release, dissemination, or impacts" [1]. This thesis will focus more on the Radiological and Nuclear aspects (since it is developed in the framework of a Nuclear Engineering Master Thesis project).

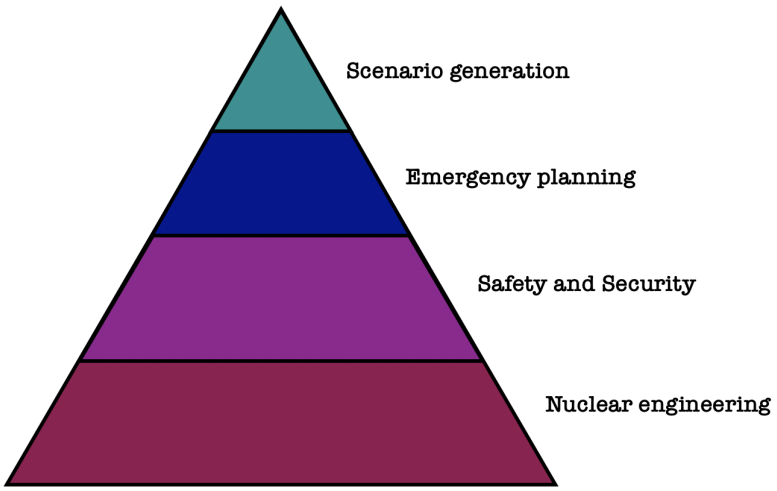


Figure 1.1: scheme depicting the overall context where this thesis is located

In this context of preparedness and emergency planning, training programs and exercises represent a crucial aspect. The building and development of scenarios is a cornerstone in preparedness training, because it allows the personnel in contact with critical substances to regularly review key concepts of security training. In addition, emergency training includes several thematic areas such as guidance, best practices and raising awareness on security in general. As it can be seen in figure 1.1, scenario generation is a part of a more complex picture, composed of emergency planning and security training in the context of nuclear engineering.
Traditionally, training scenarios are built exclusively manually, which is a method that presents some drawbacks, such as:

- *low scalability*: this is a consequence of the fact that the process itself has multiple bottlenecks, if the number of operators involved in scenario development is increased, the development speed won't increase accordingly.
- *low variability between scenarios*: due to the strong dependence on the scenario developer skills and knowledge, if the same person is tasked to develop more scenarios, a point of low variability between scenarios will be reached eventually.
- *time-consuming tasks*: the process includes repetitive and time-consuming tasks, which can lead to bottlenecks and low variability.
- *non-standardizes formats*: there are no universal formats for the scenario drafting, and this could increase confusion and insurgence of unexpected post-development work.

Having in mind these limitations, an helping tool that could aid experts overcome such drawbacks during scenario development can be of big importance. The main concept is the development of a *natural language processing* program to automatically generate training scenarios as an assistive tool for emergency planning.

1.2 Objectives

The main objective of this thesis is to investigate the contribution of Large Language Models (LLMs) in the generation of realistic and plausible scenarios in the context of emergency training. After having explored the potentialities and drawbacks of Natural Language Processing (NLP) tools, the aim is to develop a multifaceted understanding of the implications of employing such models in the context of scenario generation. First, the previous material present in the literature will be analysed, in order to develop a broader comprehension on the subject. Then the most relevant tools will be explored, focusing on the realization of certain evaluation criteria, defined after a research of open-source scenarios present in public databases. Finally, the effectiveness of the proposed solution will be addressed, in the form of expert-based evaluation.

1.3 Significance of the study

This thesis works towards having significant and relevant implications for the nuclear security field, thanks to a novel approach brought about by the integration of state-of-the-art natural language processing technologies. Exploring the usage of LLMs in this field has the potential to drastically improve the approach of experts and policymakers in the context of emergency planning. In fact, findings might contribute in streamlining the process of scenario development, increasing the heterogeneity of training scenarios and helping in the standardization of a scenario language, thus resulting in a boost in the preparedness in the face of nuclear threats.
Shifting the focus on the Natural Language Processing field, this research makes its contribution by increasing the knowledge corpus, with a showcase in real-world challenges.
As the world delves into the research of increasingly variegate solutions for evermore complex security problems, exploring and understanding such promising tools may prove of central importance.

1.4 Thesis Organisation

In chapter 2 a literature review in the context of Artificial Intelligence (AI) and Natural Language Processing will be reported, with a focus on the contribution of Large Language Models. Chapter 3 will report a brief but technical overview of the main components used in Natural Language Processing, with special focus on the Transformer architecture. Then, chapter 4 will be devoted to the methodology and tools used to develop the workflow of the thesis, comprising a full description of scenario generation and evaluation phases. Next, chapter 5 will report the main results obtained during the evaluation phase performed by expert scenario developers. Finally, chapter 6 will summarize the main conclusions drawn after analysing the obtained results.

Chapter 2

Literature Review

In this literature review, the aim is to assess the potential impact of employing Natural Language Processing tools (in the form of Large Language Models) in the framework of developing training scenarios for security in critical infrastructures, being careful of identifying useful insights and valuable conclusions.
Nuclear infrastructures can be subject of external threats that could lead to dangerous consequences to human health. In order to preserve their important role, it is strongly recommended to advocate a systematic and risk-based approach on security. This represents a basis for the design and evaluation of physical protection systems against a design basis threat for the sabotage or theft of radioactive or nuclear materials [2, 3]. Since all these systems include people for their correct implementation, training process of involved personnel represents a critical aspect in the pipeline [4].
During recent times, safety training has seen important developments in the implementation and in the instruments and methodologies adopted. New technologies, such as virtual reality, have been used to improve the effectiveness and spread of such training procedures [5, 6]; while other emerging tools (like LLMs) can potentially constitute an important piece of the puzzle. With this review, content generation applications for LLMs will be explored, in order to explore similar work cases and meaningful insights on LLMs' use.

Research Questions

Having the aim of constructing a complete but relevant analysis, the following Research Questions (RQs) will be investigated.

RQ 1) In what fields can examples of LLM applications for scenario development be found?

RQ 2) What would be the impact on the scenario development process if an LLM is employed?

RQ 3) What criteria should be used to evaluate the quality of a scenario?

RQ 4) What ethical considerations or potential risks are involved in the utilization of an LLM for scenario generation?

Answers for these questions will be useful for the development of the workflow needed in this project. Previous work will be used as a reference for key aspects, such as specific implementation of the model or selection of evaluation modality and criteria.

2.1 Impact of LLMs in AI

A Large Language Model is a probabilistic model capable of dealing with natural language (in our use case exemplified by English) for general-purpose applications. Its functioning leverages on common patterns occurring in natural language text, while making use of complex software architectures [7] to achieve a deep understanding of the meaning of the sentences. Such tools reached a high level of performance, making them usable for complex applications such as text generation, summarization and information extraction [8]. This architecture can be employed for tasks different than text manipulation, for example image or video generation using the so-called Diffusion Transformers [9].
Their nature falls under the broader concept of Artificial Intelligence, since LLM are a part of the field of Machine Learning (ML). Some light will be shed on such context before further describing LLMs.

2.1.1 LLMs and AI: an overview

Artificial Intelligence has represented a relatively recent but shocking field of research, with countless future implications and practical applications. AI relates to the use of intelligence from a program, as opposed to a human being. In this context, intelligence is related to the ability of perceiving the environment and taking actions that maximize an objective function. When AI is employed for language processing tasks, it deals with words and reasoning, thus it might offer wrong solutions, as opposed to deterministic algorithms [10]. In its core, AI also cover the capability to learn from past experience, so that the artificial agent must be able to take decisions on the basis of previous knowledge and address complexity and ambiguity.
Summarizing these concepts in one sentence, Artificial Intelligence is a trait given

to machines able to perform tasks that, if performed by human beings, would require the use of intelligence [11].

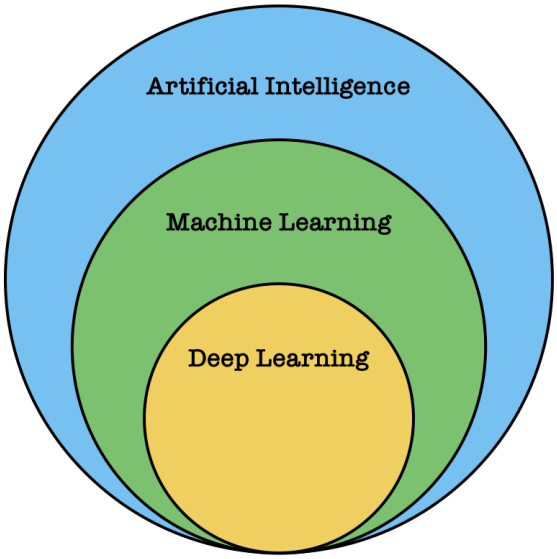


Figure 2.1: scheme depicting the role of ML in the general framework of AI, LLMs can be considered a sub-set of Deep Learning

Considering the graphical representation of figure (2.1), AI can be divided based on fields of applications. If Machine Learning is defined as a system with the ability to extract knowledge and patterns from raw data, and using that knowledge to predict arbitrary characteristics on unseen data, it follows that ML is actually a specific application of AI. AI can generally refer also to an algorithm that accepts predefined rules to predict a certain desired event, but Machine Learning models do not need to be given patterns or rules. In fact, they can increase their accuracy with an increase in the amount of data for their training phase [12]. To complete the picture, Deep Learning represents a class of Machine Learning algorithms, where the architecture is made of multiple layers (i.e. *deep*), in order to better represent high-level features of the input data.
Focusing on a more specific application of Deep Learning, the concept of Natural Language Processing can be introduced. This notion refers to a field of computer science and linguistics that involves the modelling of natural language using algorithms and statistical tools [13]. NLP systems are able to process, analyse, understand and even produce human language in human-readable text (in form of lists, messages, programming code snippets and more [14]).
Since the beginning of history, natural language has been the main way through

which *homo sapiens* has established communication between members of his species [15]. The ability to manipulate such natural language by artificial agents (i.e., machines and computers) promises enormous and revolutionary opportunities, but Natural Language Processing is a field that still needs deep study and a cautious approach [16]. After the Second World War, Alan Turing proposed a possible solution for the quest of assessing whether a machine is able to think. This test, originally called Imitation Game [17] by Turing and later known as Turing test, is an experiment where an interrogator (human) must provide written questions to two players (one is a machine and the other is a human) and assess which one is the machine. A machine is said to pass the test if the interrogator could not tell reliably the difference between the two players.

2.1.2 Impact of LLMs in the field of AI

In the years after Turing's work, important progress has been made in improving and perfecting such intelligent machines. The recent uprising of Large Language Models has marked a significant breakthrough in the field of Natural Language Processing and, more generally, in the framework of AI, with concerns about biases, environmental footprints and socio-political impacts [18] in different fields, for example medicine [19], creative engineering design tasks [20], application development [21] or patient simulation [22]. LLMs, for example with models like GPT-3, have demonstrated remarkable proficiency in understanding and generating human-like text, showing their ability to comprehend context and memorizing language patterns [23]. These models, as stated before, make use of the principles of Machine Learning, where algorithms learn patterns and make predictions based on vast amounts of data [24]. What distinguishes LLMs is their capacity to generalize meaning from large data of written text, allowing them to respond like a human being to a wide range of queries.
This abrupt surge in LLM capabilities has given rise to discussions about ethical implications [25], biases, and responsible use of AI systems in various applications, from chatbots [26] to education [27]. The field of LLM-based applications has been growing significantly and numerous studies are emerging, with focus on their construction, applications and evaluation strategies [28]. Despite, or maybe due to, this scientific rupture, LLMs face several challenges and potential drawbacks, such as practical and ethical concerns in the use of language models in education (ranging from technological readiness, replicability and privacy) [27]; or the challenges in corpora and vocabulary size, together with the complex structure of language in a LLM [29]. Additionally, editing an existing model can prove to be challenging, because the aim of effectively altering the behavior of the model can collide with new, unexpected and negative habits [30]. Considering several applications for LLMs (such as chatbots, computational biology, computer programming or creative

work...), it emerges a strong need for relevancy and error correction, minimization of biases present in pre-training datasets and model interpretability [31].
As these language models continue to evolve, the impact of AI in common tasks and technological applications will become more and more pervasive, potentially reshaping traditional relation with either creative and repetitive work in ways never seen before.

2.2 Applications of LLMs in content generation

Evolution of Large Language Models is an important area of research, routing from traditional statistical models, passing by neural models, and arriving to the most recent developments of pre-trained LLMs [32]. Such models have demonstrated extraordinary performance in natural language processing tasks, thus leading scientists to effectively employ them in multiple and diverse applications [33]. This section aims at researching literature with the objective of finding answers for RQs 1 and 2.
One of the most critical limitations of LLMs employment is the relatively narrow context window, that allows for generation of restricted content per each call. Hierarchical architectures and other strategies have been developed in order to address this issue, with examples ranging from screenplays generation [34, 35, 36] to procedural game levels design, employing nested model calls [37]. More in-depth implementations can be explored, for example ones considering auto-regressive self-attention in order to create latent representations to maintain sentence level semantics [38]. Additionally, multiple transformers and image generation models can be chained together, in order to produce consistent and easily understandable story plots [39].
Selection of the generation approach is strictly dependent on the model dimensions. In fact, a bigger model can deal with a greater degree of complexity, thus reducing the need of human intervention, while smaller models require more intermediate steps for a satisfactory deployment. With the recent introduction of deep neural networks, several applications rose towards more data-driven approaches, for example table-to-text generation [40, 41] or automatic report generation [42]. If data is limited, then LLM's real-world adoption can be difficult, thus specific model architectures need to be designed. Specifically, multiple steps in the process can represent a possible solution, as proposed by Chen et al. [43], where first, a content selection procedure is adopted, then language modeling is used for composing coherent sentences, constructed acquiring prior knowledge.
Furthermore, the lack of sufficient data makes the generation of collaborative stories creates unique challenges for Natural Language Generation (NLG) tasks. For such reason, works like the one of Du and Chilton [44] assume an important role for

LLM text generation. In their work, if a proper dataset of human generated stories is constructed, then an instruction-tuned model (trained with the before mentioned dataset) can introduce robust improvements in performance.
Several fields previously monopolized by human intelligence and creativity are now interested by the rise of LLMs, ranging from theatre script to table-to-text generation. Automatically generated contents are then processed in an evaluation phase, where their quality can be assessed by either human or automatic evaluators. This research showed that problems with the context window and lack of data are particularly crucial. Coherence within a long text turned out to be an important challenge in the frame of content generation (which includes scenario development). Despite that, several implementations, with nested calls or gradual content extraction can prove to be effective in facing such challenges.

2.3 Improving and evaluating LLM output quality

When content has been generated by the model, it is of paramount importance to define critical evaluation criteria and evaluate its goodness with respect to identified criteria. This section aims at answering to RQ 3, by searching in the literature what are the most used evaluation criteria and methods for NLG.
Recent research is showing a peak in LLM-related papers, and those related to evaluation of outputs are no exception [45]. This is a consequence of the increasing potentialities and use-cases of Large Language Models, shifting from conventional NLP tasks (such as sequence tagging) to more human centered tasks (like content generation for email writing or general structured text) [46]. With this in mind, several frameworks with the aim of aligning LLM responses with human needs have been developed; starting from Reinforcement Learning from Human Feedback [47], to Preference Ranking Optimization [48]. Such algorithms have been developed in order to improve the evaluation reliability or to reduce the amount of training needed to achieve the same result, thus improving the overall efficiency. Several other techniques exist to cope with these purposes (for example SALMON [49], Black-Box Prompt Optimization [50] or Tuna [51]), but more complex tasks require the use of an expert-based component in the evaluation pipeline, as exemplified in the work of Mirowski et al. [34].
In order to evaluate Natural Language Generation, traditional, fully-automatic methods such as n-gram overlap between outputs and references are increasingly less satisfactory, due to the increase of LLM potentialities in creativity and diversity based tasks [52]. For such reason, prompting and fine-tuning LLMs to evaluate other LLMs have been recently proposed and will be further explored [53]. Chiang and Lee showed that there is consistency between LLM evaluation and human,

expert-based evaluation, with a certain stability over the prompt formatting [54]. Such analysis was performed on scoring method for evaluation, which is a procedure where models like GPT-3.5 and GPT-4 achieve an accuracy comparable to human labels for translation tasks [55]. Nonetheless, it remains of high importance to compare automatic scores with human-based evaluation, in order to demonstrate if a particular metric compares well with human judgement [56].
Another evaluation methodology for complex tasks consists in comparing two or more different generated texts and choosing the better one. For such methodologies, ChatGPT has shown better performance with respect to previous evaluation metric, despite showing biases and inconsistencies [57]. If models with moderate size are employed for this task, their performance will be better than just prompt scoring [58].
Other methods consist in a Boolean scoring for metrics that require Yes or No answers, like for example grammar [59], faithfulness in summarization tasks [60] or factuality of generated text [61].
Evaluation usually targets specific aspects (story generation, dialogue, summarization, etc.), and the definition of key criteria for such tasks is essential. Such criteria are usually taylored for the specific application, but also automatic, LLM-based, auto-calibrating evaluators can be employed [62]. This process involves the use of examples rated by humans to automatically generate evaluation criteria. Such criteria can be also automatically reviewed, for example to eliminate ambiguity, by properly querying another LLM [45].
After this analysis, it can be deduced that human presence is still needed in the evaluation process. Fully automated systems fall short in providing consistent and reliable quality estimations, thus a human-in-the-loop approach is paramount to ensure human-criteria consistency. The choice of the evaluation criteria are strictly dependent on the specific implementation, thus a thorough analysis of the context and the objectives of the application is needed. However, some basic criteria traditionally used in NLP applications, such as BLEU (machine translation), ROUGE (text summarization) or perplexity can be employed to have a first assessment of the output quality. For a more in depth analysis, human-aligned criteria, such as Truthfulness and Clarity, are usually preferred when dealing with more complex applications such as long content generation.

2.3.1 Learning outcomes for personnel

The scenarios will be employed as a training strategy to effectively convey security awareness on first responders and critical infrastructure personnel. Such strategy, in order to be functional, needs to clearly state what learning outcomes are expected to be acquired by the training players. These learning outcomes have been structured, in order to comprise multiple threat scenarios in a Radiological-Nuclear (RN)

framework [63]. The effects of security training information have been analysed in the framework of radiation protection to optimize teaching time or to increase the flexibility for the participants [64].
Automation of scenario generation can shift the focus of the developer from the most tedious steps to truly important aspects, such as the learning outcomes regarding personnel. This implies a more reliable framework, where the quality of the final product gets important benefits.

2.4 Ethical considerations and risks

The use of Large Language Models involves a necessary analysis on the ethical risks and implications, which are closely related to their implicit and structural drawbacks and limitations. Models are observed to capture and reproduce harmful content contained in their training datasets, thus amplifying occurrences of toxic language usage or social biases [65]. The aim of this section is to find satisfying answers for RQ 4.
LLM related risks are multidisciplinary and range on different specific areas of harms and hazards: Discrimination Hazards, Information Hazards, Misinformation Harms, Malicious Uses, Human-Computer Interaction Harms and Automation, Access, and Environmental Harms [66]. LLMs are found to be beneficial to security and privacy in some domains, while also posing important vulnerabilities, such as user-level attacks due to their human-like interaction [67]. Additionally, privacy risks represent a critical aspect of LLM utilization, since embeddings can be reverse-engineered to disclose sensible information contained in training data [68]. Demographic biases represent another important risk about LLMs, thus a robust evaluation pipeline is needed for their assessment, which can be achieved by relying on syntactically diverse prompt structures [69].
Moreover, there is a need to measure and reflect on ethical issues inside the evaluation phase. In fact, this procedure benefits greatly from the contribution of human practitioners, which can be tasked to evaluate a broad range of impacts, exemplified by the terms *fairness and inclusion* [70]. This study showed common themes among participants, related to the evaluation process, identification of NLG goals and quality criteria. Selection of interview sample, however, can prove to be limited to a certain group of people, thus leading to conclusions which might not reflect the assumptions of practitioners working in other fields.
LLMs introduce important potential risks regarding their usage in various fields. In the frame of CBRN security and scenario generation, an important aspect is related to the privacy of the data employed. Training LLMs require hundreds if not thousands of examples to obtain noticeable behavioral changes, but such data can be unavailable, due to the fear of potential data leakages.

2.5 Gaps in the literature

After surveying the literature, examples of LLM applied on CBRN security training scenario development couldn't be found, thus revealing a context gap in existing literature. Similar workflows have been explored, as seen in this review, ranging from theatre script generation to game design. Having in mind the importance of security drills, this work is proposed as a first step towards merging generative models with Radiological and Nuclear security training, thus hopefully paving the road towards deeper and broader analyses.

Chapter 3

Background and Theoretical Framework

LLM(s) represent an AI framework that can deal with multiple languages (either human or programming languages) and can perform complex operations such as next word prediction, content generation, text summarization or language translation. Their usage rose extremely quickly in the past few years, due to their ability of analysis and language comprehension, and their extreme flexibility in terms of applications and use cases.
Recently, the evolution of the number of papers containing the keywords *Large Language Model* and *Language Model* has seen an increase in the number of published papers, on the subject of LLMs, from 0.40 per day to 8.58 per day [32].
LLMs' way of functioning is based on a rich history of improvements and problem solving solutions, which in the end produced the so called Transformer architecture. These models evolved from simpler tasks such as next token prediction (a *token* is a commonly occurring chunk of text, like a word or a part of it) and simple, short sentences generation, to outstanding insights and emerging abilities such as:

- **In-Context Learning**: if the user provides a prompt that contains one or more task demonstrations to the LLM, then the model is more likely to generate the correct answer.
- **Instruction Following**: with this term we indicate the ability of the LLM to follow instructions for new tasks even without explicit examples. This proves a much improved generalization ability.
- **Step-by-Step Reasoning**: if the user prompts a model with a so called Chain-of-Thought prompt (which basically means a prompt that mirrors human problem-solving methods, where a bigger problem is decomposed in

multiple smaller steps), then the LLM will be more likely to solve complex problem that requires intermediate reasoning steps.

In this chapter, a brief overview of LLM components with their basic functioning is presented. But before entering into the details of how an LLM works, a brief introduction on how a Large Language Model is able to understand meaning is presented from a higher-level perspective.

High-level functioning of LLMs

The main objective that needs to be accomplished is to "teach" the model the meaning of words, and to understand it. In order to do that, a sufficient quantity of relevant and high-quality data (composed of books, internet archives or articles) must be fed into the model. This huge training corpus has the purpose of teaching commonly occurring patterns inside language, so that the model can learn (using probabilities) what could be the next word given a chunk of a sentence. But how does the model manipulate words, if it can only deal with numbers and basic arithmetic operations? The answer lies in the *embedding* concept. This term refers to an indexed numerical vector able to store some meaningful data. Practically, this means that each word is assigned to a specific numerical vector (with often hundreds or thousands of dimensions) able to encapsulate its semantic. These embeddings will be then grouped in the embedding space, which is a multi-dimensional space containing all the word vectors in such a way that similar words are grouped together spatially.
This concept, paired with very large dataset training, enables the model to perform word arithmetic, which means basic operation capable of manipulating the meaning of the words. A common example of this concept is represented by the following equation:

$$E_{king} - E_{man} + E_{woman} = E_{queen} \tag{3.1}$$

Which means that if we extract the embedding for the word *king* from the embedding space, we subtract the embedding for the word *man* and we add the *woman* embedding, we obtain the embedding of (or a vector very closely related to) the word *queen*.
At this point, the trained model can understand inter-words relations (if a word is very distant from another, then their meaning is probably different and poorly-related). Now a comprehension of patterns must be injected in the model in order to perform the task of next-token prediction. Another wording for this task is the following: "based on the patterns observed in the training data, what could be the most probable token following the user query?". To answer this question, a model needs a training process, in which it can "study" patterns inside large text

databases. This process consist of giving the model a sequence of tokens (like a part of a sentence), and asking what it would predict as the next token (or a word). This token is masked, so that the model isn't actually able to see it, but is known in general. The objective that the model needs to accomplish is to predict the correct word (which means that it must be equivalent to the word present in the full sentence of the training database). This goal is translated in mathematical terms by a loss function, which will be minimized if the model predicts the correct word.
This was a high-level explanation of the functioning of a Large Language Model, in the following sections a deeper analysis of their parts and their general integration is presented. The first component that needs some dive-in is the Artificial Neural Network, that represents the starting element from which LLMs evolved.

3.1 Artificial Neural Networks

A neural network is a web of connections between computation units, which are conveniently called neurons, due to the resemblance of this structure to the human brain. This structure is able to identify patterns in a given input (which may be an image to classify or a text to analyse or to extend) and provide an output able to generalize such patterns. This way of functioning is made possible thanks to a training phase were data is fed to the network and the weights that connect neurons are updated in order to minimize a certain, pre-defined loss function.
This technique has become popular for clustering and classification tasks, where the most famous examples are image and text classification, not to mention several everyday applications such as medical diagnosis, marketing or product recommendation. Further refinements and complexity increases have broadened the framework for Artificial Neural Networks, improving their properties of self-learning, adaptability and I/O mapping capabilities. This has boosted their performance on more complex applications, such as Natural Language Processing, data processing and analysis, robotics and content generation (either image or text) [71].
In the following part of the section, more details will be presented on how a neural network works and what role does it play inside an LLM.

3.1.1 History and architecture

In their core, Neural Networks can be thought as large mathematical expressions that try to make a prediction or an approximation from a given set of features. These models try to achieve something which is very similar to regular linear regression.
Modern Neural Networks derive from the work of Frank Rosenblatt on the development of the *Perceptron* [72].

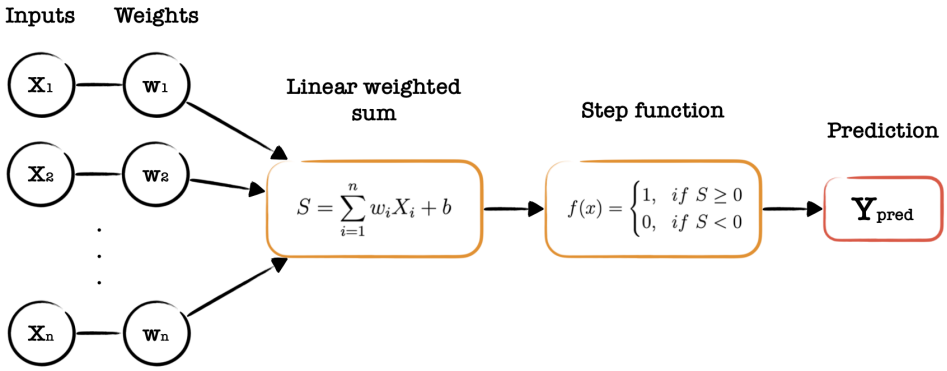


Figure 3.1: diagram of the *Perceptron*

Figure 3.1 represents the perceptron algorithm, a basic sequence of calculations used for binary classification problems. Focusing on the elements from the left, let's start by considering the features of input data, represented by the X_i components of the figure. They will then be multiplied by the *weights* (w_i), which are the coefficients that the model aims at optimizing during the training. The linear weighted sum will be corrected by a bias term b. The final computation is the classification done through the activation function (in this case it is a simple step function that gives 1 if the value of the sum is greater or equal 0, and 0 in the other case).
The method used for the optimization of the weights is the gradient descent algorithm, for which the simplest example is the *delta rule*:

$$\Delta\omega_i = \alpha(t - y)X_i \tag{3.2}$$

Where the term on the left represents the numerical change for the value of the *i-th* weight of the perceptron, t is the true value and y is the predicted value, X_i is the *i-th* feature and α is the so called *learning rate*, which represents the step size of each iteration of the minimization problem. Equation 3.2 represents the case where the activation function is a step-function. This means that the derivative of the activation function is not included, because it would have value 0 everywhere except at 0, where it does not exist. This is a simplified version of the backpropagation algorithm.
This algorithm is very rarely used nowadays, that's because of the presence of better performing algorithms and the fact that the *perceptron* is a linear classifier, which means that it will perform poorly with data that is not linearly separable.
If we want to overcome this issue, we must introduce a superposition of layers in the so called multi-layer perceptron. This consists of several layers of perceptrons (having non-linear activation functions) joined together to form a fully connected

grid. This grid represents a network of artificial neurons, thus forming a Neural Network.

Architecture

As stated before, Neural Networks are composed of several nodes, each divided into different layers. These layers are divided into input layers, output layers and hidden layers, located in between the input and output. The hidden layers are called in this way because they cannot be observed either from the input and from the output, despite being essential in the model functioning. These layers, in fact, introduce some complexity in the patterns that the network is able analyse, thus expanding the applications to non-linear mapping functions.

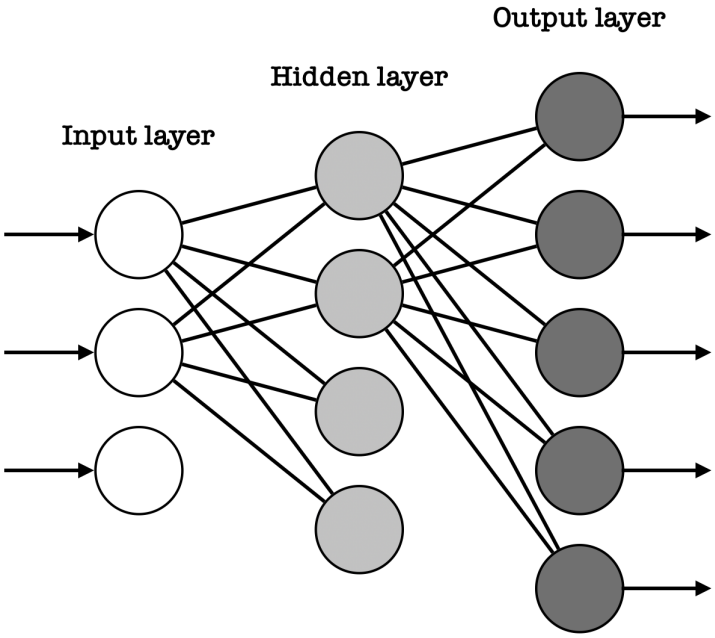


Figure 3.2: general scheme of an Artificial Neural Network with 1 hidden layer (some connections are not depicted for ease of representation)

The input layer consists of neurons that manipulate the input values. They can be just nodes (so they don't perform any calculation, a simple example is the multi-layer perceptron) or perform active calculations (like Convolutional or Recurrent layers). Input layers accept numeric tensors as input data, which can represent text, images, videos or other quantities. The size of the layer is a *hyperparameter*, meaning that it needs to be specified before the training process.
The hidden layer, as said before, introduces degrees of complexity in the patterns

that the model is required to recognize. The higher the number of hidden layers, the deeper is the Neural Network, the bigger is the time the model requires for the training process. Additionally, a very deep network can tend to overfit (see fig 3.4) more with the training data.
Finally, the output layer is composed of the neurons whose output represents the final prediction. Its dimension depends on the type of the problem that the user wants to solve: for example, if it is needed to solve a binary classification task, a single neuron with 0 or 1 as possible outputs is sufficient.
If we consider a network where all neurons have connections, we call that network a Fully-Connected Neural Network. The simplest architecture is where information flows only through a single direction (from input to output). We call these networks *Feedforward Neural Networks*.

Convolutional Neural Networks

Convolutional Neural Networks (CNN) employ convolutional layers at the input layer, in order to compute a filtering approach on a input matrix. This operation is composed of a dot product between a filter and the input matrix, and repeating this process by sliding the filter over the components of the original matrix.

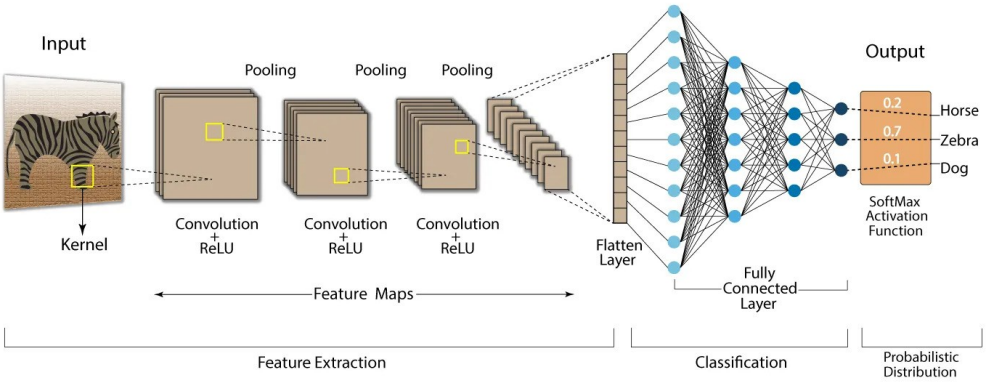


Figure 3.3: general architecture of a Convolutional Neural Network[1]

After this operation a CNN uses as an activation function the ReLU transformation, which introduces non-linearity to the model. Then, a convolutional layer is able to extract features from the input data and provide a *flattened* (which means pooled in a 1-dimensional vector) representation of the data. This simpler representation

[1] `https://medium.com/codex/understanding-convolutional-neural-networks-a-beginners-journey-into-the-architecture-aab30dface10`

is made possible thanks to the *pooling layers*, which downsample the data to reduce dimensionality.
The final component of a complete CNN is a Fully-Connected Neural Network, which can perform regression or classification tasks. These Networks are a special kind of Feed-Forward Neural Networks, which are employed when the data to be processed has high dimensionality, such as image processing or natural language processing.

Recurrent Neural Network

Dealing with natural language represents a complex task, mainly due to the fact that meaning is not usually a straight-forward concept to grasp, even for single sentences. The analysis of meaning requires knowledge on single words but also on overall context, thus sequential manipulation of input text is needed. This can be done if the output of the network is fed again inside the network, allowing the model to keep memory of past instances. This type of network is called *Recurrent Neural Network* (RNN). Such networks are able to update an internal variable to process sequences of inputs. This capability makes them best at processing arbitrarily long sequences of text and predicting the most probable next work. An important drawback of this architecture is that, in order to better predict words, more weight is given to recent information. This means that RRNs are not able to memorize data for a long time and they tend to forget previous inputs.

Long-Short Term Memory

RNNs can incur in some relevant problems in their functioning, one of which being the *vanishing gradient*. When training RNNs with gradient based approaches and backpropagation, each weight of the neural network receives an update proportional to the partial derivative of the error. During the whole training process, the value of the gradient can become vanishingly small, thus reducing drastically the information contribution of some layers and slowing down the learning process. *Long-Short Term Memory* (LSTM) [73] is an architecture made specifically for dealing with such limitation.
This model is able to remember information for long periods of time by default. It does this by introducing a state cell that is subject to modifications from the LSTM, which is developed in such a way that it is able to decide if new information is relevant and can be added to the previous knowledge of the model. In short, LSTM is made of a forget gate able to determine what information is relevant and should be kept from previous steps, an input gate which decides what information is relevant only from the current step, and the output gate determines what the hidden state should be for the next step of the network.

3.1.2 Training and backpropagation

Training refers to the process where the model learns specific patterns extrapolated from the data. In order to achieve this, a function that acts as an information error to be minimized is required. This function is called *loss function* and its purpose is to calculate how far the model's predictions are from the true data. For example, if the model is required to generate the next token in a test sentence and it makes a correct prediction, then the loss function will be low; if instead the model generates an incorrect token, the loss function will be high. This loss value will then be used for the *backpropagation* phase, where the loss function is used to find the weights and biases that led to the wrong prediction and update their values based on the loss magnitude. This calculation is done through the *gradient descent* method, which substantially calculates the gradient of the loss function and adjusts the weights and biases to reduce the loss.
This can be achieved by different strategies, such as Momentum, RMSprop or Adam.

LLM training

Globally, considering the deployment of an LLM, the entire process of training involves 3 main steps:

- The first one, called *pre-training phase*, involves the gathering of a large and diverse dataset from the internet, so that the model can learn a broad spectrum of language patterns. This data corpus needs to be cleaned in order to remove unwanted information and noise. Then, after cleaning, text needs to be tokenized. Then the model is trained to predict the next word in a sequence of text, in order to be able to generate human-like text. After this step, the LLM learned how to predict the next word of an input sequence just like a human, but is not able to follow instructions or answer questions.

- In order to make the model able to follow instructions and produce usable outputs, an additional training step is required, called *instruction tuning*. In such process, the model is provided with the user's message as an input and an example response as an output. Then the model learns how to generate responses by minimizing the difference between the predictions and the desired outputs. In this way, the model can understand the intended instruction of the user and to retrieve the necessary knowledge from its database to follow that instruction.

- After the last step, the model is able to answer questions and follow instructions, but it may also provide harmful or incorrect answer. With the aim of making the model Helpful, Honest and Harmless (HHH), a last step of the training

process can be implemented, which is commonly referred to as Reinforcement Learning from Human Feedback (RLHF). Its functioning starts from the generation of multiple outputs starting from the same prompt, then a human labeler ranks the answers from best to worst. This data is then fed to another NN which acts as a reward model, able to understand human preferences. Such reward model is then capable of replace human beings in the ranking process and further fine-tune the LLM in order to align it with human preferences.

Overfitting

Overfitting is a common problem encountered during the training phase. During the training phase, a Machine Learning developer could think to optimise the parameters in order to maximize as much as possible the performance of the model with the training data. This means to train on the maximum amount of data available, and relying on the fact that such corpus is perfectly representative of the possible working-points of the model. This strategy, although it might seem reasonable, makes the model perform particularly poorly when the model elaborates some unseen patterns or deals with data sufficiently different from the training corpus. This, in essence, comes from the extraction of residual variations in the training data (i.e., noise) on the model parameters.

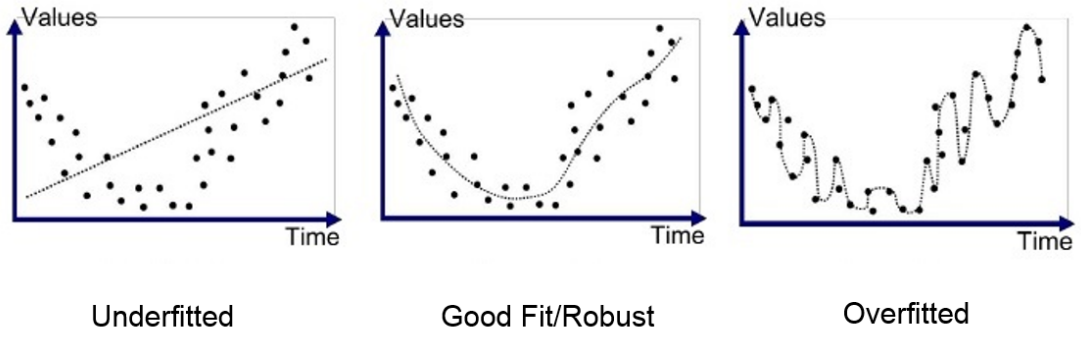


Figure 3.4: simplified graph depicting the difference between overfitting, underfitting and just-right representations of mathematical models[2]

In order to avoid this, a technique called *regularization* can be introduced, where, in order to improve the robustness of the model's representation, the model is

[2]https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76

encouraged to be more flexible. Some examples of regularization are dropout (where some random features of the model are forced to be ignored [74]) or weight-decay (introduction of a term in the loss function that increases with the size of the weights [75]).

3.2 Transformer

In their paper, Vaswani et al. [7] introduced the architecture of the Transformer, designed to work using parallel computing from GPUs. This model was proposed to leverage the higher efficiency of the *attention* mechanism, able to capture long-term contextual information using GPUs. This way of working is more computationally efficient than using Convolutional or Recurrent NNs. Its original application was for natural language translation, and involves an encoder followed by a decoder. Here, figure 3.5 depicts the general architecture of the Transformer, from a higher level point of view. The two main elements, the Encoder (left) and the Decoder (right), can be distinguished, along with the components that define their functioning.

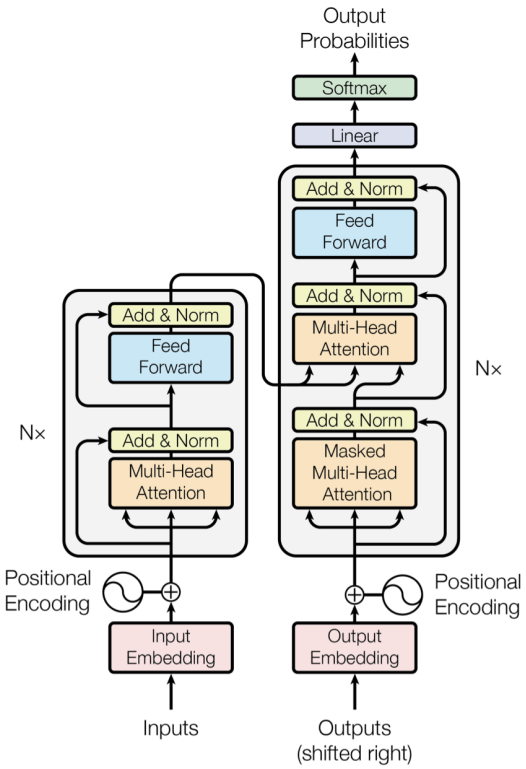


Figure 3.5: architecture of the Transformer model [7]

After a closer look at the picture, it can be seen that, starting from the Encoder part, the inputs are embedded and encoded with their positional information. This means that an input sequence of words is transformed into token vectors. Such vectors are then joined together into one single vector, but in a way that preserves the meaning of the original sentence, thus it must keep also each word position information. This operation is performed by the positional encoding. In the original paper, positional encodings have the same dimensions as the embeddings, thus they can be simply added to the embeddings. Positional Encoding (PE) vectors are calculated by using sine and cosine functions of different frequencies [7]:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \tag{3.3}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \tag{3.4}$$

In the previous equations, *pos* refers to the position of the token, d refers to the size of the token embedding, i refers to the sequence of the vector. After this step, the processed inputs are fed to the Multi-Head Attention layer (more on this component in section 3.2.1), and finally connected to a Feedforward NN. The ultimate role of the Encoder is to map an input sequence into an abstract representation that encapsulates all the retrieved information from that sequence. Inside the encoder we can see that the outputs of each component are added to the original input sequence, in the so called *residual connections*. Such connections have the role of allowing gradients to flow directly through the network, thus helping the model "reminding" the representation of the original state. Layers are then also normalised, in order to stabilize the process in case gradients assume very low or very high values.
The original architecture proposed by Vaswani et al. was composed of a stack of 6 identical Attention layers, either for the encoder and for the decoder.
The Decoder is very similar to the encoder, where its ultimate purpose is to generate predicted text sequences. The main difference with the Encoder is the presence of another sub-layer that processes the output of the Encoder with a Masked Multi-Head Attention. This Attention sub-layer is masked in order to process in parallel multiple instances of the input data. The following Multi-Head Attention module that connects the Encoder with the Decoder will make sure that the encoder input sequence is taken into account together with the decoder input sequence up to a given position. Practically, a Decoder starts with a *start-of-sequence* token and the entire output of the Encoder. It then computes the probability of the next token of the output sequence, and starts again the process starting with the previously generated token, until a special *end-of-sequence* token is reached.
The final part of the Decoder is composed of a linear layer and a Softmax function. The linear layer (composed of a fully-connected NN) acts as a classifier, outputting *logits* (quantities related to probabilities) that identify words according to a learned

dictionary. The Softmax function turns those logit scores into probabilities, thus allowing the model to choose the word with the highest probability.

3.2.1 Attention mechanism

In this section we will describe with more detail the core of the Transformer model, i.e. the *Attention mechanism*. This block allows for a word at a certain position to be influenced by all the other preceding words, independently from the distance. Without this component, simple encoder-decoder architectures would encode the initial input sequence with just a single vector, thus causing a potential loss of information (fig 3.6).

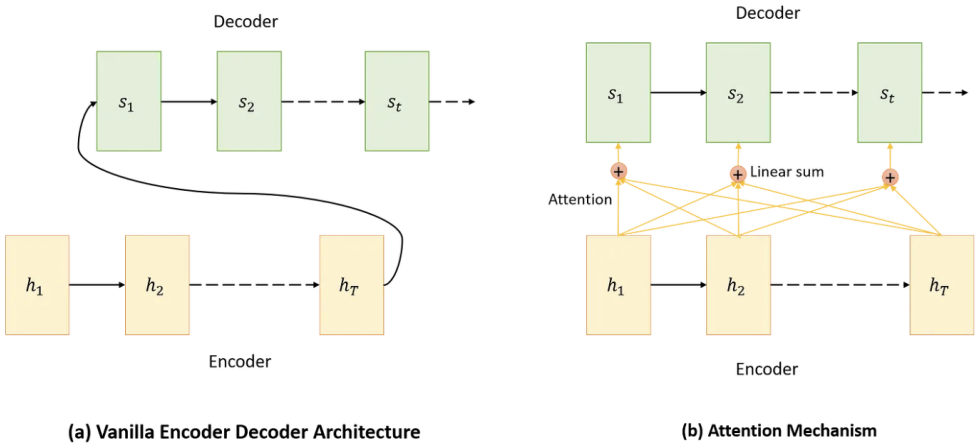


Figure 3.6: Difference between an encoder-decoder architecture with and without Attention[3]

Attention based architectures consider all the hidden states of each encoder at each time step, allowing them to make predictions on which should be the most informative. Such functioning is made possible thanks to a neural network trained to learn which hidden state retains more information.
The attention neural network will keep changing its output based on the time step, thus changing the attention given to the input sequence over time. Practically, this component computes 3 linear projections of the input embeddings, obtaining the Key (K, associated with candidate words), Query (Q, associated with the

[3] `https://shashank7-iitd.medium.com/understanding-attention-mechanism-35ff53fc328e`

analysed word) and Value (V, associated with the recommended word) vectors. Such Q, K and V concepts are related to retrieval systems. Q and K values are then compared with each other to obtain attention weights, which finally determine the corresponding Value values with high attention weight. Often it can be referred to as Self-Attention, due to the fact that instead of focusing on different sequences, it attends only the input embedded sequence. For the sake of computational efficiency, Attention is usually implemented with multiple parallel blocks, which have their own Q, K and V values. Such blocks are called *heads*, which explain the name of Multi-Head Attention.
The variant employed in LLMs performs the same calculations but it masks the vectors after the current token, in order to prevent the model from "looking forward" when predicting the next word. For this reason it is commonly referred to as *Masked Multi-Headed Attention*. This practically means that the sequence is divided into chunks, where the preceding tokens are added to 0 and the following ones added to $-\infty$. In this way, when computing probabilities with the Softmax, all $-\infty$ terms will be cancelled out.

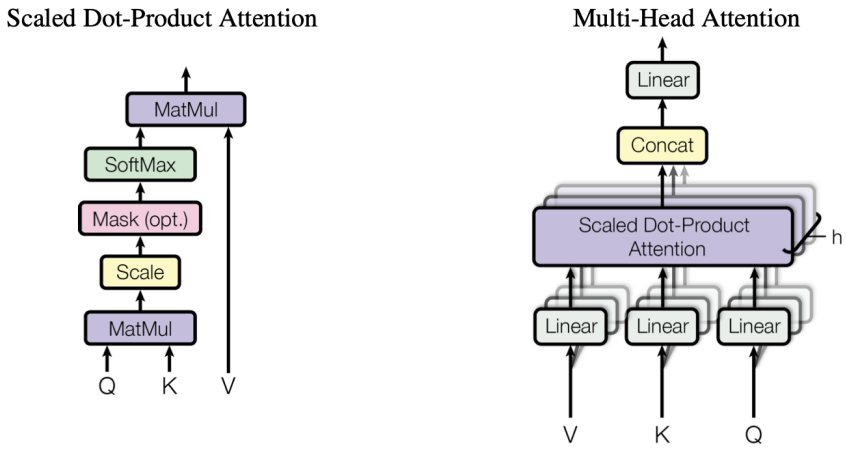


Figure 3.7: detail of the Scaled Dot-Product Attention and its Multi-Head scheme [7]

Chapter 4

Methodology

In this chapter an overview of the methodologies used will be presented, along with the final workflow of the proposed solution. Such workflow is composed by a hierarchical implementation of multiple queries and answers from an LLM, which will then be compared based on different prompting techniques. A set of evaluating criteria was chosen, in order to assess if the overall quality of the scenario is acceptable. The evaluation phase was then performed leveraging the expertise of human scenario developers, tasked to assess if generated scenarios follow the selected criteria.

4.1 Tools employed

This section will present an overview of the tools and programs used for the development of the workflow.
The main programming language used during the development of the project was *Python*[1]. This language was chosen due to its huge documentation, ease of use and widespread application. It is developed using an open source license, which means that it is freely usable, and its deployment in Machine Learning and Deep Learning applications is extremely consolidated. The proposed implementation was developed in a Google Colab[2] environment. This represents a web service for interactive computing which requires no program installation (other than that of the internet browser) and allows free access to cloud-based computational resources, including Graphical Processing Units and Tensor Processing Units.

[1] `https://www.python.org/about/`

[2] `https://colab.google`

This implementation used as a Language Model the `gpt-3.5-turbo`[3] family of models. The reasons for which this model was chosen are its broad range of applications and its large dimensions (since it stores 175B parameters). Its API is also freely available from the OpenAI website, after the user subscribes for a free account.
Another tool that was explored during the development of the thesis is the *LangChain* framework[4]. It allows users to build applications, leveraging LLMs, that are context-aware and manifest reasoning abilities. It is an orchestration framework that manages dependencies and interactions between chains of agents. Such agents are represented by an LLM which chooses a sequence of actions to take, so that it acts like a reasoning agent.
With the aim of developing an user-friendly and effectively taylored evaluation form, the *HTML*[5] markup language was chosen, along with *JavaScript*[6] programming language. Such tool is commonly employed for displaying content in a web page, allowing the developer to create a document with a customized structure.
Before describing the implementation of LLMs for automatizing scenario generation, in the following section the general structure of a training scenario will be presented.

4.2 Reference scenarios under consideration

In general, training scenarios are built starting from a review of the performance objectives for the program. Threats and hazards are then extracted from the risk assessment study. Then, availability and capability of incident (or accident) stabilizers (for example personnel, equipment or systems) are assessed, together with public emergency services (police, medical or fire fighting departments). Existing regulations are then addressed, which might be specific for the reference facility. Consequently, proper emergency response plans are developed, which should be specifically connected to the hazards and threats present. From here on, personnel is trained through drills and exercises in order to practice and refine the emergency plan.
A balanced emergency plan must address realistic accidents, hazards and threats, it must be realizable and complete in the covering of the important issues. The process of planning is recursive, since it requires continuous modifications, and alive, thus a re-evaluation is needed in order to identify, correct and re-assess critical parts of the training plan. This means that emergency planning is a never-ending

[3] `https://platform.openai.com/docs/models/gpt-3-5`

[4] `https://www.langchain.com`

[5] `https://developer.mozilla.org/en-US/docs/Web/HTML`

[6] `https://www.javascript.com`

procedure, keeping in mind that the most important aspects are the people and the "big picture", in order to give a reasonable assurance that the scenario is able to cope with deviations from the original plan [76].
The IAEA [77, 78, 79] and EU CBRN preparedness programs [80] archives were examined in order to find some reference scenarios. The reason for which only open-source materials was employed is to be able to produce a consistent and versatile study. This allows to simplify the process of potential future improvements and developments.
All participants and partners, either from local to international level, must coordinate in order to develop and accurate planning program. The pipeline of a good quality scenario should accomplish certain specifications, in order to make the final product complete and the process reliable. Consequently, the following list (extracted from an OECD document for developing nuclear emergency exercises [81]) can be addressed as a series of planning requirements:

- Identify overall exercise objectives and specific sub-objectives.
- Develop terms of reference for planning groups and technical support, etc.
- Agree on the overall scope and limitations of the exercise.
- Agree on the specific level of participation required, including decision-making mechanisms and/or involvement of decision makers.
- Determine the participation in the planning process, including relevant stakeholders.
- Determine the participation in the exercise conduct, including the full range of identified stakeholders.
- Determine the evaluation mechanisms.
- Identify the financial and resource commitments.
- Determine the level of realism required.
- Determine the overall progression of events, the timeline of the scenario and duration of the exercise play.
- Ensure that the roles and responsibilities of players correspond to their actual functions.
- Determine the strategy for informing the public about the exercise.
- Identify a structured planning process, recognising the amount of elapsed time that it will take to complete each phase.

- Identify and consider the possible follow-up actions that may be required.
- Acknowledge the roles and responsibilities of participants, and decide on whether they need to be independent or not (e.g. players as evaluators, etc).

After extracting a total of 5 scenario samples (always keeping in mind the previous list of requirements), some core components, that each scenario should include and describe, were identified:

- **Title**: it quickly conveys the principal characteristics of the scenario.
- **Location**: it should be consistent with the user query and the context. Additionally, a brief description of its main characteristics should be provided.
- **Participants**: this element is referred to personnel who participates in the scenario and their role. Additional members like public, media or local police can also be involved.
- **Difficulty**: this metric represents the general difficulty and is related to the nature of the threat, the presence of multiple concurrent dangers, etc.
- **Principal threats**: these represent the main threats that participants should tackle:
  - release of a radioactive source, should be consistent with the difficulty and the context;
  - additional threats (like fires, explosions, blackouts, victims). Might be optional and should be consistent with the difficulty and context;
- **Learning outcomes**: these components are essentially the scenario objective, they represent what the participants will learn after taking part in the training.
- **Timeline**: general description of the narrative and detailed timeline with events, time stamps and actions. It should also include some injections that controller must enforce during certain sections of the training.
- **Consequences**: what will be the consequences of the scenario and at what level:
  - population health and on involved personnel;
  - indirect disruptions on authorities and infrastructures;
  - societal, political and economical level.

Such elements represent the main characteristics that a scenario should include and describe. In order to assess if the quality of the generated scenario is sufficient, an evaluation form was developed, in order to send it to experts in scenario generation for an expert-based evaluation process. Such form is based on some key criteria selected before the generation phase, in order to define some useful guidelines that could orient also the work-flow (chapter 4.5).

4.3 Prompt Engineering

In the development of this section, the contents where informed by the principles and concepts covered in the *ChatGPT Prompt Engineering for Developers* course from *DeepLearning.AI*[7].
Prompt engineering is a discipline whose aim is to develop, optimize and improve the prompts that will be submitted to the model [82]. It is often referred to as the "*art of asking the right question*". It is employed by general users, developers and researchers to make the model generate an answer that adapts as much as possible to the actual user's intents, which may vary from question answering to analytical reasoning. A useful example for this practice is asking a query to Google search and modifying it to increase the probability that the response matches the user intents. Several examples in the literature show the performances of the different prompting techniques of medium and large sized models, testing on diverse benchmark datasets [83, 84, 85].
Before analysing further this approach, it is important to differentiate among two general kinds of models: a base LLM and an instruction-tuned LLM (as was introduced before in chapter 3.1.2):

- BASE LLM: these kinds of models are only capable of predicting the next words based on the training data. They cannot follow instructions or answer to questions if these tasks are not present in the training data.
- INSTRUCTION-TUNED LLM: these models are trained to follow user instructions, and try to be Helpful, Honest and Harmless. This process is achieved by a fine-tuning process of feeding of input-output pairs that include instructions and responses that follow those instructions. Reinforcement Learning from Human Feedback (RLHF) can then be employed to further refine the model. It makes use of human labels to tune the model responses towards user-aligned responses [47].

[7] `https://learn.deeplearning.ai/courses/chatgpt-prompt-eng/lesson/1/introduction`

It follows that an instruction-tuned LLM will be more prone to give a satisfactory response, if the query is adequately designed with respect to the instructions. If the user wants to extract the highest value from the model answers, then effective prompting techniques are one of the best options. Also, direct fine-tuning of the model parameters can be explored as a valid alternative, but often it is not an available technique, due to the requirements of sufficient high-quality data and relatively high computational power.

4.3.1 Prompt engineering techniques

These procedures are a set of good practices that allow users to leverage most of the LLM capabilities. Such practices are based on adapting and changing the wording inside the prompt, so that the answer from the model is more probable to be satisfactory. In order for the prompt to be of high quality, it is good practice to follow the following principles:

1. The instructions should be clear and specific. Often times, this doesn't mean that the prompt should be short, but its length must be appropriate to convey all the relevant information.
   Some good tactics for this are:
   - usage of delimiters to clearly indicate distinct parts of the input
   - asking for structured outputs, since a sufficiently big model can follow a format provided in the prompt
   - asking the model to check if conditions are satisfied
   - strategies like Few-Shot Prompting, this concepts relates to the practice of giving the model one or more examples of a good response already in the prompt.
2. The user should give the model time to think. This principle suggests that the user should consider the model as another rational being that needs physical time for a good answer.
   Some good tactics for this are:
   - specify the steps required to complete a task in the prompt
   - instruct the model to work out its own solution before rushing to a conclusion.

Commonly, model implementations (a very widespread example is OpenAI playground[8]) include a parameter called *system message* which accepts a textual input

[8] `https://platform.openai.com/playground`

from the user. This message is then passed to the model each time it processes a query, and acts as an instruction to tell the model how to interpret the conversation, its profile, tone and posture. It is intuitively similar to a "whisper" that the model always hears.
Having in mind these concepts, in the following an overview of the strategies adopted in this project is presented:

- **Zero-shot prompting**: this prompting technique is the simplest and most straight-forward to implement. It is solely based on asking the model to perform a certain action, without providing examples to further instruct the model. In the framework of this project, it is used as a reference for more complex prompting techniques.

- **Prompt chaining**: this technique allows for breaking the main tasks into sub-tasks, in order to streamline the overall problem. The LLM will be prompted with a sub-task, and then its answer will be sent as a parameter to another prompt, and so on. This process can be repeated indefinitely, until the first, complex task is addressed with a satisfactory performance. Such technique, apart from letting the model achieve a better solution, allows for an increase in reliability and controllability of the implementation.

- **Chain-of-Thought prompting**: Chain-of-Thought (CoT) prompting technique allows for complex reasoning capabilities using intermediate reasoning steps [86]. The variant proposed by Kojima et al. [87], which essentially involves adding the wording "*Let's think step by step*" before each answer, was employed. This process allows for more correct responses, and perform better if it leverages the potentialities of larger language models.

- **Tree-of-Thoughts prompting**: Tree-of-Thoughts represents an additional generalization with respect to Chain-of-Thoughts [88]. This process enables the model to reason and self-evaluate the intermediate thoughts made towards solving a problem. A simple wording addition inside the prompt allows the model to accumulate knowledge progressively and, at the same time, rectify the errors autonomously [89].

- **Few-Shot prompting**: when dealing with more complex tasks, few-shot, 2-shot or 1-shot prompting can enhance in-context learning capabilities of LLMs. This technique is implemented by providing the model with demonstrations (in this case first with 1, then with 2) in the prompt [90, 91].

Before describing the main architecture of the implementation proposed for the problem, in the following sections a brief overview of the unsuccessful concepts and implementations is presented.

First a tentative solution using smaller models (in particular GPT 2) will be described. Then a more complex program using *LangChain* and *RAG* approach will be shortly described. Finally, the proposed hierarchical architecture will be analysed.

4.4 Architecture of the model

Within this section, the selected procedure adopted for solving the problem will be presented. Before proceeding with the proposed model, a brief overview of the unsuccessful attempts, together with valuable insights learned from their limitations, is presented in appendix B.
The chosen architecture takes inspiration from the model *Dramatron* described by Mirowski et al. [34]. Dramatron consists of multiple instances of an LLM, leveraging prompt chaining and structural context techniques.
The specific implementation that was adopted for this thesis uses a sequence of prompt messages and system messages to generate important information in a sequential way. This partition allows the model to focus only on the relevant parts in sequential order. The single parts were then joined together to create the final scenario, allowing for a longer output and a more complete and reliable generation. However, such methodology introduces some rigidity in the process, since the output is forced to follow the predefined sequence.

4.4.1 Structure of the architecture

First of all, having the aim of reducing the requests to the model as much as possible (since OpenAI gives a limit of 200 daily requests, together with a maximum of 3 requests per minute for the free tier subscription[9]), the recurrent information were generated first as a batch. Such recurrent information was then used by the successive model requests to generate a timeline and the possible consequences, varying the prompt technique employed. Each time the model was called for generation, the parameters and chat history where re-initialized, in order to reduce the chance of biases. Then, the necessary context information was passed at the prompt level, in order to be able to evaluate the desired portion of the general pipeline.
Such general information is composed of the scenario title, the location, a general description of the context, people involved, the radioactive source (if any), the difficulty, the threats and the learning objectives. This procedure was not employed

[9] `https://platform.openai.com/docs/guides/rate-limits/usage-tiers?context=tier-free`

for the zero-shot reference model, since this model is simply asked to generate a scenario, without providing further instructions. The results of such overly simple procedure will be used as a baseline for comparisons in the evaluation phase.
Then the prompting techniques described in chapter 4.3.1 were employed in order to generate a timeline, controller injections and possible consequences. The prompting techniques under consideration were:

- *Half-shot*: this consists in the injection of a general structure (chapter 4.2) inside the prompt, without giving specific examples. This technique gives only the structure to the model, without focusing on content.
- *Chain-of-Thought*: in this case the model was specifically prompted to "think step by step" in order to provide a thought process for another model call, tasked to generate the actual scenario based on the thought process generated before, and the general scenario structure. This techniques gives the model the ability to perceive a perspective of the general task.
- *One-shot*: in this case the model is provided with an example from the human generated scenarios reported in chapter 4.2. Each example refers to a timeline, some injections and possible consequences, along with a reference prompt for generating such scenario elements.
- *Two-shot*: similar to the previous case, but the model is provided with 2 examples, extracted from the same database.
- *Tree-of-Thought*: this prompting was addedd to all the previous models in order to extract guidelines for improving the scenarios. The specific wording was extracted from Hulbert [89], in order to simulate a dialogue between 3 experts evaluating the scenario proposed. Their reasoning was then used to ask the LLM to generate a modified version of the original scenario, keeping track of the guidelines of the 3 simulated experts.

These elements were then joined together, in order to output the complete scenario. The following picture shows a graphical representation of the general functioning of the generation techniques, each arrow represents a model call which will generate some content.

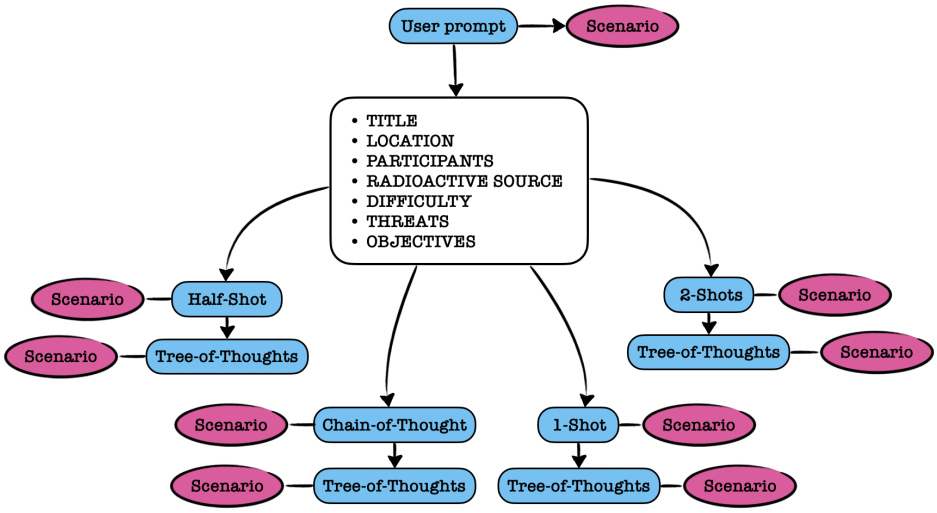


Figure 4.1: high-level scheme of the architecture for the scenario generator, each arrow represents a model call (or multiple calls)

This procedure led to the generation of 8 scenarios, plus a reference one where the basic prompt was employed as the only input, for a total of 9 scenarios.
Scenarios provide a general overview of the main components before starting the exercise itself, which are a title, a location, a brief description of the context, a list of all the participants with their roles and supposed tasks, a radioactive source, the level of difficulty, the main threats (such as possible fires or unexpected issues to be addressed during the scenario) and the main objectives, which represent what participants should learn after taking part in the scenario. After this, previously described prompting techniques are employed in parallel in order to compare their outputs, in terms of consistency and relevance with previous elements and with the original user intents.

4.5 Evaluation methodologies

Evaluation phase is extremely important, since the outputs of language models are probabilistic (which means that the same prompt could produce a different output every time). This means that the quality level of the output needs to be measured, and this is done through its evaluation. In order to correctly assess this evaluation phase, several techniques could be employed, each with a different performance based on the desired application.
After explaining the chosen method, the selected metrics under investigation will be presented.

It is important to highlight the fact that evaluating LLMs is still a topic where substantial research is needed. Specific methods and procedures used during this phase might change over time, according to the progresses made in developing auxiliary tools or in the development of the specific application under evaluation.

4.5.1 Automatic evaluation

This approach is characterised by prompting another LLM, by giving it the desired instructions and asking it to give a verdict, to evaluate the output.
Although this may seem a circular and biased process, it should be remembered that human intelligence has always been used to evaluate human intelligence (for example in job interviews, exams or college finals). Using AI systems to evaluate AI systems has the important potential of automating a task that requires too much effort and resources (time, money and personnel) to be carried out only by humans.
Automatic evaluation can limit the time and cost constraints of having a human evaluating the answer, but as regards the accuracy and quality of the evaluation, it can perform noticeably worse in accuracy and reliability. Even though a model is capable of giving good evaluations on a specific task, it is not sure if it may be able to generalize on different tasks. Moreover, the limitations regarding subjectivity are still a concern, because the evaluation's outcome is dependent on several external factors, such as the model's parameters, the training data, its biases etc.

Evaluation metrics

Some common practice metrics can be employed during the automatic evaluation phase, here a brief overview is presented [92]. Such metrics can refer to multiple Natural Language Processing tasks, from sentiment evaluation to semantic understanding, from summarization to question answering. Usually, in such cases, a reference text is available for comparison analysis of generated responses, and metrics serve the purpose to assess the distance of the model answers from the reference text.

- **Accuracy**: this metric is used as a measure of the correctness of a model answers based on a given task. Its exact definition varies on the specific application. In order to estimate it quantitatively, some metrics can be employed:
  - *Exact Match*: as the name says, this quantity evaluates whether the model's output precisely matches the reference answer. It can range from 0 to 1, where 0 means perfectly un-matching answers, and 1 means perfect match.

  - *F1 score*: this metric is used for binary classification tasks, where the F1 score is defined as:

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{4.1}$$

    Where precision is the fraction of True Positives vs. Predicted Positives, and recall is the ratio of True Positives vs. all Positives.
  - *ROUGE*: this metric is used for text summarization and machine translation. This score measures the similarity between the generated and reference summaries using overlapping n-grams, which represent word sequences that appear in both texts [93].

- **Calibration**: when a model has to calculate a prediction, it outputs the estimated probabilities of the possible outcomes. Such estimated probabilities (or confidences), should represent the actual frequencies of the events that it is predicting. If there are discrepancies between such distributions, the model needs calibration, in order to make its predictions more accurate and reliable. A commonly used metric for calibration is Expected Calibration Error (ECE), which can be considered as an approximation of the difference in expectation between confidence and accuracy [94].
- **Fairness**: this metric relates to equal treatment of different groups of people, without discrimination of race, gender, age and more. Trustworthiness can be assessed using several metrics and benchmarks, a comprehensive and thorough example of which is DecodingTrust [95].
- **Robustness**: robustness is a metric which evaluates the performance of a model when faced with several adversarial and challenging inputs. As regards this aspect, some benchmarks and state-of-the-art models analyses can be found in the literature [96, 97].

Such evaluation metrics represent a standard in assessing answer performances in a rather wide range of applications, but when it comes to non-standard LLM applications, then traditional metrics might not be sufficient. For example, when dealing with content generation, standard answers evaluated with previous metrics can be not suitable, since a sufficient corpus of evaluation data might not be available. For these reasons, human evaluation can be more reliable, at least for the first parts of the pipeline, where the process is more exploration-oriented and less standardized.

4.5.2 Human evaluation

This procedure is composed of asking human evaluators to address the quality of the output. An important reason for employing this method is that human

evaluation is closer to the actual application, plus it can provide more accurate feedback. This approach could be executed following different modalities, some examples of which are reported in the following list:

- **Reference-based**: the evaluator is tasked to compare each output with an ideal response that acts as a ground truth. This ground truth must be constructed beforehand, and each prompt requires a reference response. Moreover, the evaluation quality is strongly related to the quality of the ground truth, which means that if the reference response is not properly built, then the evaluation will be of poor quality.
  This approach can be useful for evaluating question answering tasks, where the response of the LLM can be fairly easily assessed either as "right" or "wrong", based on the prompt question.

- **Scoring**: the evaluator assigns a score to the answer (for example, from 1 to 9), without comparing it to a ground truth. The score can be a single value or made of multiple values, depending on the complexity of the output to be evaluated. For example, in order to sufficiently address the quality of an answer to a creative task, we might need to assign several scores based on a set of relevant metrics. The evaluation may also be a flag, so for example the evaluator can simply say if the answer was concise or not or if it was relevant with the original task or not.

- **A/B Testing**: the evaluator could receive a pair of responses and rate the best one. This approach is useful if we want to evaluate the performance on different parameter configurations of the model. This is based again on the subjective opinion of the evaluator.

This manual evaluation strategy is widely adopted in a variety of tasks, and represents one of the most explored approaches when an LLM application needs evaluation. It presents however some important drawbacks and challenges. First of all, it is very difficult to scale efficiently. This means that time and cost needed are higher than alternatives. In addition, it is usually difficult to reduce costs and time employed using this strategy alone. The second point is that results are very subjective and depend on the evaluator itself (results given by an evaluator can easily be different from those given by another evaluator). This variability is given by many factors influencing the evaluation procedure, such as expertise and biases of evaluators.
Such drawbacks are certainly relevant and significant, of secondary importance with respect to automatic evaluation ones. This is due to the fact that scenarios generated in this project are not significant in number, plus currently there is no standard procedure to automate the evaluation phase. With the aim of developing a meaningful evaluation, the human, expert-based evaluation procedure was selected.

4.5.3 Reference evaluation

Before proceeding with the submission of evaluation forms to expert evaluators, a preliminary evaluation test was performed on human-generated scenarios. During this phase some modifications were introduced, in order to make the evaluation process easier, such as a brief description for core components, and more reproducible, by asking evaluators to provide some details about their background. After conducting preliminary evaluation tests, it has been noted that the average time to evaluate a fairly complete scenario is around 10 minutes.

4.5.4 Expert-based form evaluation

An evaluation form was written in HTML and divided into 3 main sections, which are:

- **Core components assessment**: in this first part, evaluators are asked to determine if all core components are present and, if the answer is yes, assign a score from 1 (minimum) to 9 (maximum). The score is based on consistency with a general description of what should be expected from the model in each scenario element.
- **General evaluation criteria**: this section is devoted to assigning a score from 1 (minimum) to 9 (maximum) for the following chosen criteria, which have the aim of being crucial components for the specific application. They have been designed in order to be compliant with the guidelines proposed by the Federal Emergency Management Agency of the United States [98]:
  - CLARITY: high score correspond to good levels of readability and grammatical correctness. This metric, despite being more subjective [99], is definitely relevant with the case of written content generation [100, 101].
  - RELEVANCE: high scores should be assigned to answers that are closely related to the user's intents. If there are discrepancies in the intents of the user and responses of the model, then a low score should be assigned. Such metric is often important to evaluate in particular use-cases, which include black-box models and lack of access to human-annotated references [102]. In fact, when developing user-oriented text-generation applications, relevance should be considered a crucial element in the pipeline [103].
  - COMPLETENESS: high scores should be assigned to scenarios able to cover all the important core components. Factual and structure completeness are usually overlooked when analysing common NLG metrics [104], thus it is crucial to explicitly evaluate such metric. Completeness has been assessed in different fields of interest, like chat evaluation benchmarks [105] or effectiveness of LLMs as coauthors [106].

  - TRUTHFULNESS: high score correspond to realistic and truthful information in the response. If the model generates false or unrealistic answers, the score should be lower. This metric is important since LLMs tend to give unrealistic answers or write wrong information, in a flaw called *hallucination*. For such reason, truthfulness is a essential metric to address the reliability and trustworthiness of LLMs [107, 108]. For such reason, expert assessment is usually a significant and effective route for truthfulness evaluation [109].

- **Final remarks**: this last section contains open-ended questions, in order to ask for meaningful modifications to the scenarios and a meta-evaluation on the criteria that a scenario should satisfy.

Before answering previous section, a brief introductory questionnaire has been included, which contains questions about the background of the expert. Inside it, the evaluator is requested to insert an auto-evaluation of the skill level based on the Dreyfus Model [110] and the number of years of experience.

Details regarding forms

The ultimate goal of this specific evaluation implementation was to obtain a "golden standard" for the framework, in order to produce realistic and useful scenarios. Such method permits users to assist emergency planners in the scenario developing phase, so that the model acts as an assistive tool for scenario planners.
Instructions were provided to the evaluators (both in a briefing and in textual formats, see appendix C), in order to remove as much as possible the probability of mistakes or errors in the usage of the tool. Having this same objective, it was decided to move the reset button to the top, above all questions, so that an user, that wants to submit the form, is less prone to mistakenly deleting all the answers. In order to increase readability, the form was split in two columns, one containing the text of the scenario, while the other with the questions for the evaluators. A description of the core components is provided but hidden, and it can be accessed through a question mark button. This feature was included in order to focus the evaluators attention on the answer, while giving the possibility to retrieve detailed information only when needed.
The nature of this evaluation procedure implies that feedback from evaluators will be used as an input to improve further iterations of the process. Future work on the user experience can still be performed, such as introducing a progress bar, which updates every time a scenario has been evaluated.

Chapter 5

Results

This chapter will be devoted to the results obtained during the evaluation phase. First, an overview of a representative set of training scenarios will be presented, then the results and feedback of expert planners will be reported. Such feedback will represent a useful material for future iterations and updates for the pipeline. Then, limitations of the study will be highlighted, closing the chapter with an overview of ethical considerations about this specific evaluation step.

5.1 Representative scenarios

With the aim of introducing a reliable and feasible evaluation pipeline, 3 representative prompt types were chosen for 3 representative events, for a total of 9 scenarios to evaluate.
The Tree-of-Thought technique has been neglected, since it introduced minor modifications to an already prepared scenario. Effects of this technique alone could be assessed in a future work. Also the 2-shot and half-shot promptings were left behind. The first one was discarded because it was very similar to the 1-shot, however it presented a greater degree of confusion with the example data. As regards the half-shot technique, it presented similar structure to the Chain-of-Thought, yet it showed a lower amount of details.
After such preliminary observations 0-shot, Chain-of-Thought and 1-shot prompting techniques have been selected as a representative sample, so that results of evaluation forms will be used to improve and standardize the procedure. The goal of this pre-evaluation phase was to analyse, get feedback and streamline as much as possible the actual evaluation phase. This was done because, in future iterations, the evaluation pipeline can benefit greatly from a campaign where experts are tasked to evaluate not only scenarios, but also the evaluation form itself, so that valuable insights could be implemented later on. Additionally, such insights could

be used to train an automatic evaluator in order to, at least partially, automate even the evaluation phase and obtain more consistent results.

5.2 Main evaluation results

In this section, results and key insights from evaluators will be reported. Scores could be used as a quick overview about the prompting techniques and the different performances of the models, they can be found in appendix A.
The most relevant comments and suggestions will be reported hereafter. Such observations will be useful in order to improve and update future iterations of the process. Selection of the following comments was done manually, choosing common insights from multiple evaluators.

5.2.1 Hospital radioactive source theft

This section is devoted to report evaluation insights for the *hospital radioactive source theft* scenario.

Zero-shot prompting

In this section comments regarding Zero-shot prompting are reported. Evaluators showed agreement in giving low scores for the radioactive source assessment, since lack of important details is a crucial flaw in scenario implementation.

> *The description is rather generic, because the source is known within the Hospital, in principle all details of the source are available (type, radionuclide, strength...). I can imagine that it is for example a Ir-192 source for Brachytherapy?*

> *No source (which is normally known in a hospital setting, I think), and no time line. No people that got into contact with the source potentially outside of the perpetrators.*

Important core elements were found missing, more specifically timeline, injections and possible consequences after scenario completion were not present in the scenario. Moreover, security team was missing, which is critical since the scenario relates to a theft of a radioactive source. Information about other personnel involved was missing several details.
In general the scenario was found to be particularly generic, and *Completeness* score was consistently low among evaluators.

Chain-of-Thought prompting

In this section comments regarding CoT prompting are reported. Some issues were discovered regarding details about the radioactive source chosen for the scenario.

> *Little information about source, this should be known in case of a theft from a Hospital*

> *Ir-192 is used in brachytherapy but the scenario might lack information on the activity of the source.*

Such specific information is definetly relevant in the context of training scenario (since the activity of a source defines the procedures to be taken and the measures to be considered).
Another important insight regards the timeline, more specifically its duration with respect to the context.

> *The timeline is clearly not realistic for the tasks foreseen. I would argue that the timeline should be extended to at least 8 hours*

Usually the model outputs timelines with total duration of about 2 to 3 hours, which is considered too short for this particular context.
Another insight regards the absence of radiological monitoring personnel, whose task should be the monitoring of radioactivity levels in case of releases or contamination of radioactive material. In the case of a spurious release of radioactive materials, radiation safety personnel must act in order to assess potential dangerous areas for human health.
Generally, scenarios generated using this prompting technique were found to be much more detailed and consistent with the proposed scenario structure. Important details are still missing, such as a more in depth description of the radioactive source.

One-shot prompting

One-Shot uses an example in order to instruct the model how a proper answer (given the query) should look like. When implementing this technique with the scenario generator, multiple criticalities were found. The first relates to the actual timeline:

> *[The timeline is] inconsistent with user prompt: it switches to roadside accident*

This resulted in a consistently low score for the *Relevance* metric. Such mistake was probably due to confusion of the model between the specific prompt and the user example, even if the model was explicitly tasked to ignore information present in the example and use it only for structural relevance. Such inconsistency has a negative impact on the other metrics since some evaluators found the scenario to be not clearly readable.

5.2.2 Nuclear research center radioactive release

This section is devoted to report evaluation results for the *nuclear research center radioactive release due to operator error* scenario.

Zero-shot prompting

In this case, evaluators were concordant in acknowledging the scenario as very incomplete and lacking of crucial information.

> *The scenario is very vague and need further description.*

Details about the source and scenario location were for example missing, thus making the scenario not usable as is, in the frame of assisting scenario developers.

> Was the model realistic and truthful in the response?
> *Isotopes and quantities (or if the aim is to not provide this information, explicitly say that this information is not available at this stage for instance). How the release actually occurred (is it released to the atmosphere or to the water discharges). The type and number of players.*

> *Location of the facility would be helpful (flatland, hilly terrain, mountains, next to sea) as well as closest settlements*

Such missing elements resulted in an overall low score regarding *Completeness*, thus, further instructions are again necessary in order to make the model more relevant and consistent.

Chain-of-Thought prompting

Overall, scenario completeness increased with respect to the previous prompting technique, but consistent lack of detail in several aspects remained ingrained. As an example, radioactive source was present along with its activity, but its physical

form and way of release are not present. For this particular case, source activity is realistically too small to be released from a reactor core melt-down, but too high for a single source in a hot-lab for nuclear medicine applications.

> *The isotope (Cs137) is mentioned together with the activity (1000 Ci) but it is not said in which form it is. To me the type of source (also from where is it coming) is not clear: is it in gaseous, liquid or solid form? Is is dispersed in the air or in water discharge?*

Moreover, timeline showed to be more consistent with respect to Zero-shot, but definetly too short in time to achieve the defined learning objectives.

> *The time foreseen in the timeline together with the inject are not realistic (this is also valid for the objectives for a scenario of 2h)*

One-shot prompting

In this case, One-shot proves to confuse the model to the point where the scenario timeline is completely inconsistent with the user query.

> *The scenario is a mixture of a release at a nuclear research centre (up until the timeline) and an accident between a truck and a train (starting from the timeline).*

This behavior is probably related to the fact that examples of the timeline inside the prompt are too lengthy for the model. Such examples tend to confuse the model, even though they are provided to the system message and not to the user prompt for the specific step.

5.2.3 Street transport accident involving radioactive source

This section is devoted to report evaluation results for the *radioactive source street transport accident* scenario. This scenario is very similar with the example provided in the One-shot prompt example, so it is expected that the model stays relevant with the query. Despite this, a crucial aspect for this case is the lack of diversity in the creation of scenarios, in the sense that the model might behave "lazily" and stick too much with the example provided.
This procedure is biased by the example provided, in fact the LLM is already given the "correct answer". Nonetheless, it is still relevant to analyse since the prompt explicitly instructs the model to ignore the content of the example and follow only

the structure. If model behaves as intended, it might prove to be beneficial in a future framework where models are used to modify already written scenarios to improve diversity in scenario development.

Zero-shot prompting

Also in this case, Zero-shot prompting proves to be capable of generating only general-purpose scenarios, with multiple elements missing and an overall lack of detailed information. In fact, no information about the source is present, there is no timeline and some key participants are completely missing (like the driver of the truck).
The scenario is still relevant and truthful, but it misses several important components that are necessary to be addressed for a deployable (or close to deployment) scenario.

Chain-of-Thought prompting

Chain-of-Thought showed again a good overall performance, with some inaccuracies in some details and some struggles in maintaining an overall consistent description of the scenario.

> *What I don't like is that it doesn't understand the difference between a 'lost source' and a 'radioactive release' –> it doesn't make sense that a solid source should disperse (unless it breaks in a thousand pieces). I feel it's either a lost source or a radioactive release, not both.*

The above observation refers to a subtle mistake that the model made. It might seem a slight oversight, but it's actually relevant with respect to the scenario implementation, since knowing if the source was dispersed or not leads to completely different plans of action.

> *[...] first and last injections come very early (during scoping of the scenario) and very late (almost time for debrief).*

This observation relates to a common problem of LLMs when they try to give an answer which is meant to be as complete as possible, which is excessive verbosity. This refers to a particularly lengthy output, whose content could be exposed using less words in a similarly effective way. In this case the model tries to give injections for each phase of the scenario, even if they could be not requested or even detrimental. This problem could be assessed by explicitly prompting the model not to give injects at specific time stamps, like at the beginning or at the conclusion of the scenario.

One-shot prompting

In this case, timeline and scenario in general was significantly more consistent with respect to the other cases, due to the presence of the "correct answer" inside one of the system messages. This was done, as explained before, in order to assess the capability of models to generate diverse scenarios starting from already written ones, without confusing with the provided example.
For this task, the model is actually still confusing itself with the provided example, since, for example, a train engineer appears in the middle of the timeline. For this reason, it is important to focus on this problem in future iterations, trying to optimize the prompt in order to obtain diverse but internally consistent results.

> *In the beginning only 2 bystanders are said to be involved. This number is higher later on.*

> *[...] train engineer doesn't exist and use of T0 is inconsistent between phase 1 and phase 2.*

> *First injections happens 30 minutes before T0, so no accident has occurred yet. However, leaking fuel is already reported suggesting the accident has happened already.*

Additionally, the model tends to output exaggerated or wrong information about radioactive source and potential symptoms of people that come in contact with it. This is probably still due to spurious influences from the provided example, or more simply the model is hallucinating during generation.

> *Severity of symptoms probably exaggerated*

5.3 Discussion of the Results

Obtained results showed that Zero-shot prompting was not sufficient in fulfilling the requested fields for a scenario. After this prompting technique, models presented lack of detail and consistent omissions in terms of structure components.
Chain-of-Thought represents an improvement in terms of completeness, consistency and adherence to a defined structure (since such structure is explicitly provided). This technique still presents some lack of details and components, but with a smaller degree with respect to Zero-shot. After comparing it with the other prompting techniques, CoT seems the most promising in terms of relevance and structure of

the output, while it keeps lacking some important aspects in terms of content.
One-shot makes the model confuse itself, thus drastically reducing completeness and relevance of outputs. This might be related to the length and complexity of the provided example, which is interpreted by the model with an excessive weight. In fact, models were provided with an example related to a road accident of a vehicle transporting radioactive material, together with an example prompt for generating such timeline. This proved to be insufficient since the LLM continued to confuse the original prompt with the provided example in terms of content, while in terms of structure, generated scenario and human example were very similar. It has still been decided to submit such examples to human evaluators in order to provide complete evaluation results, eventually usable for a symmetrical training of automatic evaluators. In this way, an LLM could be provided either with high and low scores examples, so that it could be capable of evaluating consistently a wide range of possible scenarios.
Outputs of the models are still probabilistic, token-based texts, thus further experiments and updates are needed in order to align output quality to human standards. Further details must be explicitly given to the model, such as more structure for people involved, or explicitly asking for isotope, activity and form of the radioactive source.
When comparing different prompting techniques, it has been noticed that structure is more important than content. This results from the observation that CoT (which focuses on structure and step-by-step reasoning) performs better than One-shot, which holds high risks of confusion for model answers with irrelevant content. For this reason, it is proposed that gaps in model knowledge are addressed using fine-tuning of internal parameters using a sufficiently big corpus of high quality data. Structure orientation, however, can be fairly easily addressed at the prompt level, depending on specific user or work environment needs.
After comparing results of the evaluation forms, it has been noticed that timeline and inject sections present multiple criticalities, mainly due to difficulties in holding relevance with the remaining components of the scenario. Furthermore, timeline generation task might be too complex and long for a single model call, thus dividing it in multiple components might prove to be beneficial for output consistency.

5.4 User Experience

In this section, feedback from the evaluators about the evaluation procedure will be reported. Since the evaluation process takes more time than expected, some evaluators suggested to assign a specific prompt seed to each expert, in order to improve the efficiency of the process. For the sake of this initial evaluation step, some evaluators were tasked to start from the bottom and some from the beginning

of the scenarios, in order to reduce redundancy of the provided results.
Some issues were found during the full evaluation experience, since evaluators struggled to understand if a file was actually been created. In fact, the form gives an alert only when a file is successfully created, while if it is not the case (which commonly occurs when not all the mandatory questions are answered) nothing happens. An alert issuing the evaluator to fill the remaining answers could be a useful feature to implement in order to improve interpretability of the form.
Another observation was made while evaluating inconsistent One-shot scenarios. Since such scenarios were perceived as obviously confusing and un-usable, an option to skip questions and address critical flaws by providing only a general purpose comment could be beneficial for developer's interpretability of the evaluation results.

5.5 Limitations of the study

In this sections, the limitations about the study (both regarding the generation procedure and the evaluation framework) will be presented.
First, regarding the scenario generation pipeline, the proposed architecture presents a quite rigid structure, in the sense that it requires internal knowledge on how it works and coding experience to be used and eventually modified. In order to make it usable for a non technical emergency planner, some additional work is needed in order to include it in a user-friendly and, to some extent, customizable environment. Additionally, only models from OpenAI GPT-3.5 family were employed, working exclusively at the prompt level. These limitations, together with a limited context window, represent a restriction in detailed outputs. This reveals that training (either from prompt level, but risking to incur in additional response confusion (appendix B), or internal parameters level) can be employed to explore the capabilities of LLMs to learn new data taylored for the specific application.
Regarding the evaluation pipeline, a limited number of scenarios were employed (while being them the most suitable for comparison since they are substantially different from each other). Additionally, only one iteration for the evaluation procedure was performed. Future iterations could bring substantial improvements regarding criteria compliance and inclusion of relevant details. Finally, evaluation was performed only using human evaluators, while automatic evaluation procedure was not explored. In order to fully implement this step, human relevant data is still needed, but one it reaches some degree of maturity it could represent a valuable feature to streamline the process of deployment for emergency planners.

5.6 Ethical considerations about the study

This section was based on the UK Statistics Authority Committee guidelines for ethics self-assessment [111].
Ethical implications of the present evaluation work start from the observation that the actual objective of the project tends to ensure public good and reduce potential harms to the public. Actual public engagement is not addressed, since the scope of the workflow is limited to a digital environment and to the working environment of SCK CEN research center. Due to the limited context of the evaluators sample (exclusively among SCK CEN workers), research findings might not reflect assumptions of practitioners working in other centers or fields of research. Data employed in this framework respects the privacy of the evaluators, since their identity is not public and their working experience assessment contains limited, although relevant, information. Consent of evaluators in performing this activity has been properly addressed. Prudence and particular attention should be put in safeguarding this aspect, in order to reduce the risk of disclosing the identity of data subjects, along with personal information of evaluators.
Access and sharing of data is a limited aspect of this pipeline, thus improvements on this aspects are considered of crucial importance in future iterations. Results of the evaluation scenarios are in fact stored in a private drive, for the sake of easiness of access and processing of the data. Additionally, due to the rather descriptive nature of this research, encouraging further expansions and approaches represents a major goal, in any case without discouraging the analysis of different practices.

Table 5.1: Ethical Concerns and Mitigation Actions

Ethical Concerns	**Mitigation Actions**
Lack of direct public engagement	Focus on ensuring public good and reducing harms for potential misuses
Limited context of evaluators' sample	Acknowledge limitations and potential biases in research findings
Privacy of evaluators' data	Ensure evaluators' identity remains confidential
Access and sharing of data	Implement improvements for better access and sharing of data

Chapter 6

Conclusions and future improvements

This chapter will conclude the study, summarizing the key insights taken after the completion of the process, relating to the research questions and evaluation results. It will also provide some proposals for future improvements and summarize the main limitations of the study. The goal of the study should be kept in mind, which was aimed at finding a valuable way of employing LLMs to (partially) automate and streamline the training scenario development process.
It first emerged that LLMs are a useful and versatile tool for a wide variety of tasks, but when complexity reaches a certain level (as the case of training scenarios) a structured approach is needed in order for the model to comply with certain defined objectives. Several ways exist of employing LLMs for content generation, and the chosen procedure for this study was a hierarchical-based architecture. This procedure was then employed for a variety of training contexts and prompting techniques, in order to assess if some implementations could reach satisfying levels. Evaluation criteria were researched and identified, in order to standardize an evaluation process usable for each produced scenario, likewise to lay the foundations of a proper discussion about the obtained results.
Results showed that employing LLMs alone to generate a fully deployable scenario is not a plausible approach yet. In fact, such models require the presence of human modifications in order to comply with key criteria, despite variation of the prompting technique proves to be beneficial (for example, explicit structure-following prompts that use Chain-of-Thought led to an increase in completeness of generated scenarios). Nevertheless, if LLMs are coupled with reliable prompting techniques and structured architectures, outputs present more alignment with human requirements. This might prove to be a reliable way towards achieving partial automation in the pipeline of scenario generation, improving scalability and

diversity of deployed scenarios.
This study presented some limitations during its implementation, mainly related to a small number of scenarios evaluated by emergency planners, or the inclusion of only one iteration of the evaluation process. Another limitation is related to the use of only one LLM, since smaller models were extremely resource intensive to run locally, while other cloud-based models weren't available during the period of project development.
Due to the preliminary and explorative nature of the present project, future work and further expansion is definitely needed. With the aim of outputting quality scenarios more consistently, the present pipeline could be expanded with the inclusion of additional content, such as exercise questions, or with improvements related to the inclusion of insights from evaluation results. Then, with progressive iterations of evaluation campaigns, automatic evaluation can be explored as an option, in the sense that LLMs could be tasked to evaluate outputs generated by other LLMs, using human examples as training. Additionally, models like GPT-4 or Claude 3 could be explored in virtue of their bigger dimensions and longer context window. In order to make the scenario generation more complete, it might be of interest to expand the context to the full CBRN spectrum, including also Chemical and Biological threats.
While LLMs have proven to be valuable and versatile tools, their employment still needs the implementation of a structured and non-trivial approach, plus the presence of human practitioners, whose aim is to orient model responses towards human-aligned criteria. This work, despite its limitations, is proposed as a preliminary but hopefully valuable approach towards automation of some critical aspects of the scenario generation pipeline.

Appendix A

Results of the evaluations

A.1 Hospital radioactive source theft

Criteria	**0-shot score**	**CoT score**	**1-shot score**
CLARITY	8 | 6 | 9	8 | 7 | 7	2 | 7
RELEVANCE	8 | 6 | 7	8 | 6 | 8	1 | 3
COMPLETENESS	1 | 3 | 7	7 | 6 | 7	1 | 5
TRUTHFULNESS	8 | 7 | 8	6 | 6 | 8	5 | 6

Table A.1: General evaluation criteria for *Hospital radioactive source theft* scenario

Core components	**0-shot score**	**CoT score**	**1-shot score**
Location	0 | 8 | 7	7 | 8 | 6	0 | 8
Context description	5 | 2 | 8	4 | 8 | 7	0 | 7
People Involved - Radiological Personnel	0 | 1 | 2	0 | 0 | 0	0 | 0
People Involved - Security Team	0 | 1 | 0	8 | 7 | 7	0 | 1
People Involved - Civil Population	0 | 1 | 6	0 | 1 | 7	3 | 1
People Involved - Facility Personnel	0 | 1 | 6	8 | 9 | 8	0 | 7
Radiological Source	0 | 1 | 5	5 | 1 | 6	0 | 6
Learning Objectives	2 | 7 | 8	8 | 6 | 8	1 | 7
Timeline (consistency)	0 | 0 | 8	1 | 7 | 8	2 | 1
Injections	0 | 0 | 0	8 | 3 | 8	1 | 1
Possible Consequences	0 | 0 | 0	4 | 7 | 9	7 | 6

Table A.2: Core components for *Hospital radioactive source theft* scenario

A.2 Nuclear research center radioactive release

Criteria	**0-shot score**	**CoT score**	**1-shot score**
CLARITY	1 | 8 | 7	8 | 7	8 | 7
RELEVANCE	5 | 5 | 8	8 | 6	1 | 4
COMPLETENESS	5 | 2 | 1	3 | 6	1 | 4
TRUTHFULNESS	9 | 8 | 7	4 | 6	5 | 4

Table A.3: General evaluation criteria for *Nuclear research center, radioactive release* scenario

Core components	0-shot score	CoT score	1-shot score
Location	1 | 2 | 6	3 | 8	0 | 6
Context description	5 | 2 | 3	2 | 7	2 | 3
People Involved - Radiological Personnel	3 | 0 | 1	0 | 0	0 | 0
People Involved - Security Team	3 | 0 | 0	0 | 0	0 | 1
People Involved - Civil Population	1 | 0 | 1	0 | 7	2 | 0
People Involved - Facility Personnel	7 | 0 | 1	8 | 7	0 | 5
Radiological Source	1 | 1 | 0	4 | 6	1 | 0
Learning Objectives	8 | 2 | 3	5 | 7	1 | 1
Timeline (consistency)	0 | 0 | 0	2 | 7	1 | 1
Injections	4 | 0 | 0	2 | 7	0 | 0
Possible Consequences	4 | 0 | 0	6 | 7	5 | 1

Table A.4: Core components for *Nuclear research center, radioactive release* scenario

A.3 Street transport accident involving radioactive source

Criteria	0-shot score	CoT score	1-shot score
CLARITY	9 | 7	8	9 | 7
RELEVANCE	9 | 7	8	9 | 7
COMPLETENESS	5 | 3	7	9 | 7
TRUTHFULNESS	4 | 7	7	5 | 6

Table A.5: General evaluation criteria for *Nuclear research center, radioactive release* scenario

Core components	**0-shot score**	**CoT score**	**1-shot score**
Location	8 | 7	6	8 | 7
Context description	7 | 3	3	8 | 3
People Involved - Radiological Personnel	8 | 1	3	2 | 3
People Involved - Security Team	5 | 1	3	5 | 3
People Involved - Civil Population	1 | 1	7	5 | 7
People Involved - Facility Personnel	0 | 0	8	0 | 3
Radiological Source	1 | 0	7	4 | 7
Learning Objectives	8 | 6	7	8 | 7
Timeline (consistency)	0 | 0	7	7 | 6
Injections	0 | 0	7	7 | 6
Possible Consequences	0 | 0	7	6 | 7

Table A.6: Core components for *Nuclear research center, radioactive release* scenario

Appendix B

Initial attempts

This appendix is devoted to briefly report the initial failed attempts to address the initial problem. First, an approach with a smaller model (GPT-2) will be presented, then, after facing several complications, follows the last approach before the proposed one. This second tentative solution aimed at employing LangChain with RAG approach to generate satisfactory scenarios.

B.1 GPT-2 implementation

In this section the attempt on implementing the program using a GPT2 model will be briefly analyzed. This approach was first selected and then explored after observing that GPT-2[1] is a relatively small model compared to other state-of-the-art models (such as GPT-3.5 or GPT-4[2]), which makes it more suitable to be implemented. This means that it can be used for free with the Python `transformers` library, thus there is no need to use an API key or to create an OpenAI account.

B.1.1 Architecture

The pipeline was based on the MarioGPT model developed by Sudhakaran et al. [37]. MarioGPT is a model based on a fine-tuned version of the distilled GPT-2 model, in which the information contained in the prompt is encoded by a BART model.
The actual coding for this strategy was definitely non trivial and presented some difficulties in the adaptation of MarioGPT in outputting scenarios in the place of

[1] `https://huggingface.co/gpt2`

[2] `https://platform.openai.com/docs/models/overview`

video game levels. The first step was to start with a simplified version, using only a GPT-2 model and fine-tuning such GPT-2 model based on previously generated data. This data was obtained through engineered prompts given to ChatGPT and standardising the output in a JSON format. Such procedure led to obtain overly simplistic but useful data for a first evaluation of the performance of the GPT2 model.
After experimenting with this configuration, it has been noticed that the performance was extremely poor for such complex application. In fact, core components present in the evaluation form were not addressed in a sufficiently accurate way, if at all. Such unsatisfying result was probably due to a very restricted fine-tuning dataset. Another aspect is the relatively small number of parameters of the GPT2 base model, which leads to often non satisfying outputs for more complex applications. Due to the complexity of finding enough relevant data to feed the model, it has been concluded that continuing with this strategy would be detrimental.
Having this disappointing but informative result, a bigger model was explored, using also a different and more versatile framework, given by the LangChain toolkit.

B.2 LangChain implementation with GPT-3.5

After the observations done in the previous section, a natural consequence was to use a much bigger model and a framework specifically designed for the specific case-study. As regards the model, GPT3.5-turbo[3] has been chosen. It represents another model developed by OpenAI and with a much higher number of parameters than GPT-2. An important drawback of using this model was that it has a bigger cost per token, thus the use of an API key from OpenAI was mandatory. As for the framework, LangChain was adopted, due to its deep documentation and widespread use. The aim of using such design was to hopefully produce a relevant and usable output. Such objective has been pursued with the adoption of Retrieval Augmented Generation (RAG) technique, which automates the retrieval of relevant information from a database, working only inside the prompt (see section B.2.1). This aspect was essential, since the LLM under investigation is not open-source.
After some preliminary experimentation with this configuration, relevant but generic results were obtained. In fact, scenarios generated by this code would often lack some core components, which would be essential for a practically usable scenario.

[3] `https://platform.openai.com/docs/models/gpt-3-5-turbo`

B.2.1 RAG technique

RAG is a technique used to increase the quality of the language model answer and extend its capacity. Its way of working can be described by considering the name itself, in the sense that the answer of the model will be generated (Generation) based on the addition (Retrieval) of prompt-relevant information (Augmented). This procedure works at the prompt level and does not change the values of the parameters of the model, so fine-tuning and other internal modifications will not be deployed. In this section its functioning will be described.

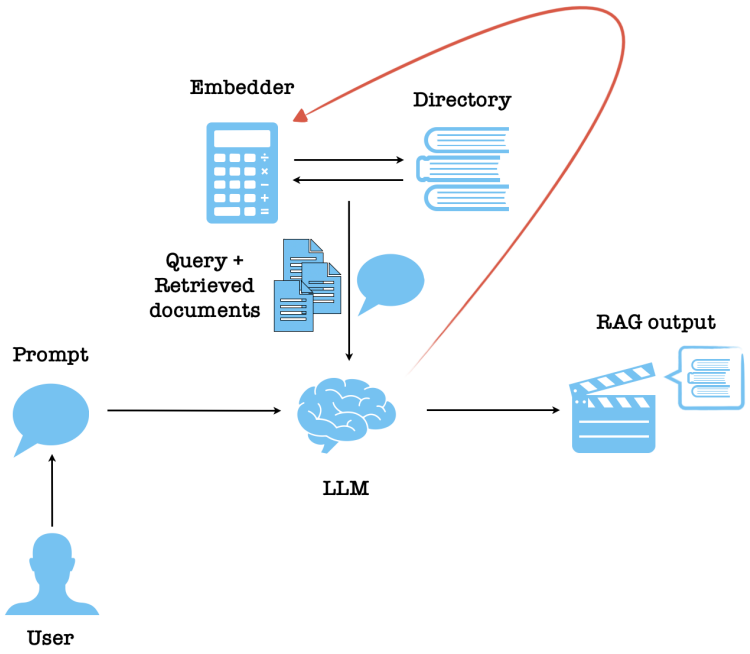


Figure B.1: general scheme of the high-level functioning of RAG

Before analysing RAG in depth, some important common problems of language models must be considered:

- LLMs can hallucinate, meaning that they can output information that seems plausible but which is factually wrong;
- if some data is not in the training corpus, then it will not be accessible for the model, except from the user prompt;
- LLMs don't have access to updated information, which might be relevant and essential for some specific applications.

RAG technique represents a way of solving these problems, without the need of training a brand new model or fine-tuning a pre-trained model, thus reducing the costs and the time involved. The only way of improving consistently the answer of the model without changing its parameters is to operate at the prompt level. If we are able to provide document snippets or chunks of text as a context inside the question, then the answer will more likely be relevant with the user needs.
Essentially, in order to implement correctly the RAG approach, first it is necessary to gather relevant text and create embeddings from it. An embedding (as previously mentioned in chapter 3) is a numerical vector that captures the meaning of a sentence or a chunk of text. A vector of numbers is needed because models can manipulate only numbers. Before assigning an embedding vector to input text, data corpus should be split into chunks. This operation is essential, because language models have a finite context window dimension and exceeding it means that the LLM cannot operate further. By splitting the context data corpus into chunks, the model can be fed with relevant information, while at the same time without overshooting the context window dimension. However, after this operation, the system should be able to distinguish which chunk is prompt-relevant. This crucial operation is handled by the *retriever*. This component of the RAG chain works as follows: it assigns a vector embedding to the user query, thus encoding its meaning in a sequence of numbers, which will then be compared with all the embedding chunks composing the data corpus. This comparison phase is done by calculating the semantic distance between the query embedding and the embedded chunks, the specific method used can vary, but one of the most common method is calculating the *cosine similarity*.
Cosine similarity is a way of ensuring that the results obtained are contextually relevant to the user prompt. This method consists in calculating the cosine of the angle between the two vectors (which means calculating the dot product of the vectors and normalising it with the product of the magnitudes):

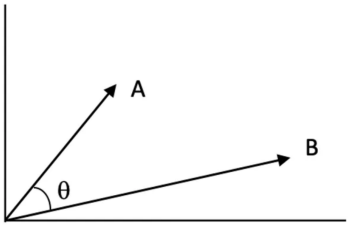


Figure B.2: two 2D embedding vectors with the angle θ between the vectors[4]

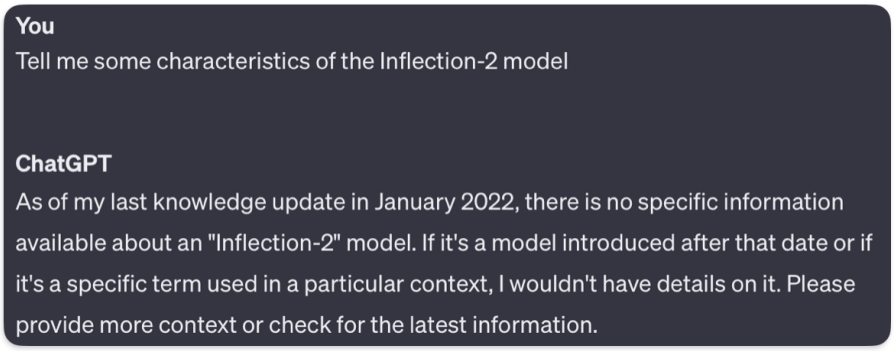
$$cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}||\ ||\mathbf{B}||} = \frac{\sum_{i=1}^{N} A_i B_i}{\sqrt{\sum_{i=1}^{N} A_i^2}\ \sqrt{\sum_{i=1}^{N} B_i^2}} \tag{B.1}$$

If this value is low, then the two vectors will be very close, meaning that their semantic meanings will be closely related.
After performing this calculation for each pair of query + vector embedding of the database, a score based on the semantic similarity can be assigned. This way, the higher the score, the closer the two vectors are, therefore a ranking list can be built. Based on this list, a certain number of chunks can be discarded, deciding to keep only the top k elements. Such chosen chunks will be the most relevant to the user prompt and can be fed to the model, thus providing a qualitative context for the response. Of course, quality is strictly dependent on the data contained in the dataset, which should be relevant with the specific problem.

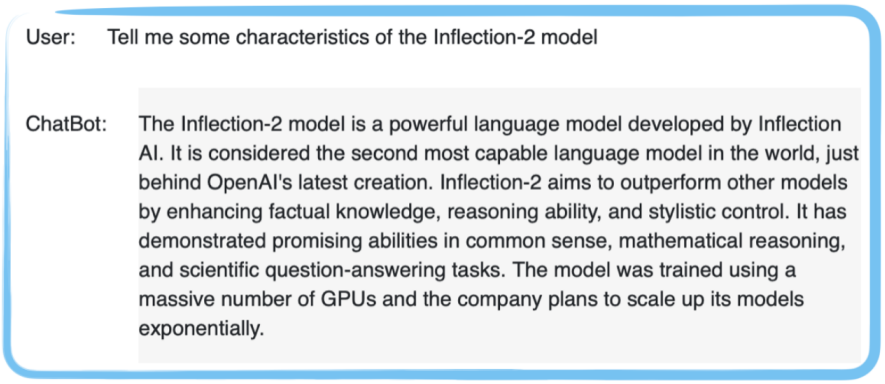
B.2.2 Retrieval from context embeddings

In order to increase the quality of the output and reduce hallucinations, a context retriever was implemented within the LLM. Such implementation included an embedder that accepts as input either a *PDF* or a *txt* file and outputs a vector database containing the numerical embeddings of the text in the input. The temperature of the LLM was set low in order to obtain a reliable and faithful interpretation of the original data.
With this feature, the model can answer questions related to more recent and specific data, provided that database includes relevant documentation about the topic. Here a simple example is considered, in which first ChatGPT (similar but more capable than GPT-3.5-turbo) is asked about a recently introduced model (figure B.3a), and the correct answer using our retrieval augmented model (figure B.3b).

[4] `https://towardsdatascience.com/cosine-similarity-how-does-it-measure-the-similarity-maths-behind-and-usage-in-python-50ad30aad7db`



(a) ChatGPT response to the query



(b) response of a model that has access to more recent data

Figure B.3: comparison between ChatGPT and a model that uses RAG, the query is the same but the accessible information is different

As it can be seen from figure B.3, the effectiveness of a model can be improved by adding information retrieval, without the need to fine-tune the whole model. In this specific case, the article[5] that the second model uses as a reference was converted into a string of text and fed to the RAG algorithm. The cited article has been published in the 26^{th} of November 2023, which is after the last knowledge update for ChatGPT (January 2022) at the moment of the writing.
This implementation proved to be useful for increasing the context of the model, but

[5] `https://www.marktechpost.com/2023/11/26/inflection-introduces-inflection-2-the-best-ai-model-in-the-world-for-its-compute-class-and-the-second-most-capable-llm-in-the-world-today/`

when it came to generate training scenarios, the performance was not satisfactory. The structure presented in chapter 4.2 couldn't be followed fully and consistently, due to the small context window of the model. In fact, the retriever was prone to extract either irrelevant or redundant information, increasing verbosity and reducing relevance inside the answer.
Thus, with the aim of having a reliable system capable of consistently generating complete scenarios, the hierarchical structure of chapter 4.4 was designed.

Appendix C

Briefing text for the evaluators

Title: master thesis "Nuclear Security: a Natural Language Processing approach" in Polytechnic of Turin.
This form is related to my work performed at SCK CEN in the context of my master thesis.
Link to fill the form:
`https://script.google.com/macros/s/AKfycbyx0e80-uW6Y8qgyKPkxL8vb4J81veHP6PbYGKt25K8P8Miv-17tXTqLUBGYRRnVBWM/exec`

Objectives:
The results of this evaluation process will be used to define and produce gold standardized scenarios which will be used to update and train models. This process is aimed at producing realistic and useful scenarios, usable by emergency planners, which means that a human is present in the loop. The large language model is an assistive tool, and does not replace human experts.

Instructions:

1. Select the chosen scenario in the left column of the form, the text will appear as soon as a scenario is selected
2. A button called "Show General Information about the evaluators" is present in the top. pressing it will open a window with 2 fields to be filled. Complete this procedure every time that the link for the questionnaire has been opened.
3. In the right column, complete all the requested questions. All the multiple answers (Present/Not Present and Yes/No) are mandatory.

4. Some fields (like the scores) are already initialized.

5. After completing the form, a submit button is present in the bottom. If pressed, an alert will appear saying that a file has been created. Such file will be automatically sent to a specific Google Drive.

6. When a scenario has been successfully evaluated, return at the top and press the reset button to reset all the answers inserted. This will give the opportunity to start another evaluation.

7. At this point, select another scenario to be evaluated and start again the procedure from point 1

Information about the study:
The scenarios are generated starting from 3 example context (hospital theft, operator error in a nuclear research center, radioactive release after a street accident), each generated from 3 different techniques.

- The first one, called 0-shot, refers to a simple query of one sentence to the model, then the scenario is directly generated without further modifications.
- The second one, called Chain of Thought, refers to a technique where the model is asked to generate a thought process before completing the task, and then after completing the thought process, the scenario is generated.
- The third is called 1-shot and it refers to a technique where the model is given an example of a human-made scenario and it is asked to generate a scenario based on the structure contained in the human example.

- the questionnaire is about the evaluation of automatically generated scenarios
- the evaluation can be performed in multiple steps or in one session
- an average time of 10 minutes has been estimated for completing the evaluation per scenario
- in total, 9 scenarios are present (3 techniques for 3 different contexts)
- additional information about the techniques can be found in the "Information about the study" section of this file

P.S. this form has some problems with Firefox, but chrome, safari or edge are fine.

Bibliography

[1] *EU preparedness and responses to Chemical, Biological, Radiological and Nuclear (CBRN) threats | Think Tank | European Parliament — europarl.europa.eu*. https://www.europarl.europa.eu/thinktank/en/document/EXPO_STU(2021)653645 (cit. on p. 1).

[2] Mary Lynn Garcia. «Design and Evaluation of Physical Protection Systems». In: *Design and Evaluation of Physical Protection Systems*. Elsevier, 2008, pp. 1–11. DOI: 10.1016/b978-0-08-055428-0.50005-1. URL: http://dx.doi.org/10.1016/b978-0-08-055428-0.50005-1 (cit. on p. 5).

[3] Alan J. Kuperman and Lara Kirkham. *Protecting U.S. Nuclear Facilities from Terrorist Attack: Re-assessing the Current "Design Basis Threat" Approach*. https://sites.utexas.edu/nppp/files/2013/07/INMM-2013-July-paper.pdf. 2013 (cit. on p. 5).

[4] Dori Ellis, John Matter, and Ruth Duggan. «Training programmes for the systems approach to nuclear security». In: *International Journal of Nuclear Knowledge Management* 3.1 (2008), p. 6. ISSN: 1479-5418. DOI: 10.1504/ijnkm.2008.018934. URL: http://dx.doi.org/10.1504/IJNKM.2008.018934 (cit. on p. 5).

[5] KwanSeong Jeong, ByungSeon Choi, JeiKwon Moon, DongJun Hyun, JongHwan Lee, IkJune Kim, GeunHo Kim, and JaeSeok Seo. «The scenario-based system of workers training to prevent accidents during decommissioning of nuclear facilities». In: *Annals of Nuclear Energy* 71 (Sept. 2014), pp. 475–479. ISSN: 0306-4549. DOI: 10.1016/j.anucene.2014.04.033. URL: http://dx.doi.org/10.1016/J.ANUCENE.2014.04.033 (cit. on p. 5).

[6] Silas Cordeiro Augusto, Antônio Carlos A. Mól, Pedro C. Mol, and Douglas Sales. «Using virtual reality in the training of security staff and evaluation of physical protection barriers in nuclear facilities». In: 2009. URL: https://api.semanticscholar.org/CorpusID:216018452 (cit. on p. 5).

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. Aug. 2023. DOI: 10.48550/arXiv.1706.03762. arXiv: 1706.03762 [cs] (cit. on pp. 6, 23, 24, 26).

[8] Jason Wei et al. *Emergent Abilities of Large Language Models*. 2022. DOI: 10.48550/ARXIV.2206.07682. URL: https://arxiv.org/abs/2206.07682 (cit. on p. 6).

[9] William Peebles and Saining Xie. *Scalable Diffusion Models with Transformers*. 2023. arXiv: 2212.09748 [cs.CV] (cit. on p. 6).

[10] W. B. Gevarter. *An overview of artificial intelligence and robotics. Volume 1: Artificial intelligence. Part B: Applications - NASA Technical Reports Server (NTRS) — ntrs.nasa.gov*. https://www.govinfo.gov/content/pkg/GOVPUB-C13-3b325954f686676ca69d9fcda90471ff/pdf/GOVPUB-C13-3b325954f686676ca69d9fcda90471ff.pdf (cit. on p. 6).

[11] Gyanendra Singh, Ajitanshu Vedrtnam, and Dheeraj Sagar. *AN OVERVIEW OF ARTIFICIAL INTELLIGENCE*. Feb. 2013. DOI: 10.13140/RG.2.2.20660.19840 (cit. on p. 7).

[12] Ian Goodfellow and Yoshua Bengio and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org (cit. on p. 7).

[13] Aravind K. Joshi. «Natural Language Processing». In: *Science* 253.5025 (Sept. 1991), pp. 1242–1249. ISSN: 1095-9203. DOI: 10.1126/science.253.5025.1242. URL: http://dx.doi.org/10.1126/science.253.5025.1242 (cit. on p. 7).

[14] Alpa Reshamwala, Dhirendra S. Mishra, and Prajakta Sharmao Pawar. «REVIEW ON NATURAL LANGUAGE PROCESSING». In: 2013. URL: https://api.semanticscholar.org/CorpusID:61858148 (cit. on p. 7).

[15] Bjorn Merker and Kazuo Okanoya. «The Natural History of Human Language: Bridging the Gaps without Magic». In: *Emergence of Communication and Language*. Springer London, 2007, pp. 403–420. ISBN: 9781846287794. DOI: 10.1007/978-1-84628-779-4_21. URL: http://dx.doi.org/10.1007/978-1-84628-779-4_21 (cit. on p. 8).

[16] Bill Manaris. «Natural Language Processing: A Human-Computer Interaction Perspective». In: *Advances in Computers*. Elsevier, 1998, pp. 1–66. ISBN: 9780120121472. DOI: 10.1016/s0065-2458(08)60665-8. URL: http://dx.doi.org/10.1016/S0065-2458(08)60665-8 (cit. on p. 8).

[17] A. M. TURING. «I.—COMPUTING MACHINERY AND INTELLIGENCE». In: *Mind* LIX.236 (Oct. 1950), pp. 433–460. ISSN: 0026-4423. DOI: 10.1093/mind/lix.236.433. URL: http://dx.doi.org/10.1093/mind/LIX.236.433 (cit. on p. 8).

[18] Dieuwertje Luitse and Wiebke Denkena. «The great Transformer: Examining the role of large language models in the political economy of AI». In: *Big Data amp; Society* 8.2 (July 2021), p. 205395172110477. ISSN: 2053-9517. DOI: 10.1177/20539517211047734. URL: http://dx.doi.org/10.1177/20539517211047734 (cit. on p. 8).

[19] Luigi De Angelis, Francesco Baglivo, Guglielmo Arzilli, Gaetano Pierpaolo Privitera, Paolo Ferragina, Alberto Eugenio Tozzi, and Caterina Rizzo. «ChatGPT and the rise of large language models: the new AI-driven infodemic threat in public health». In: *Frontiers in Public Health* 11 (Apr. 2023). ISSN: 2296-2565. DOI: 10.3389/fpubh.2023.1166120. URL: http://dx.doi.org/10.3389/fpubh.2023.1166120 (cit. on p. 8).

[20] Jan Göpfert, Jann M. Weinand, Patrick Kuckertz, and Detlef Stolten. *Opportunities for Large Language Models and Discourse in Engineering Design*. 2023. DOI: 10.48550/ARXIV.2306.09169. URL: https://arxiv.org/abs/2306.09169 (cit. on p. 8).

[21] Oguzhan Topsakal and Tahir Cetin Akinci. «Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast». In: *International Conference on Applied Engineering and Natural Sciences* 1.1 (July 2023), pp. 1050–1056. ISSN: 2980-3209. DOI: 10.59287/icaens.1127. URL: http://dx.doi.org/10.59287/icaens.1127 (cit. on p. 8).

[22] Siyuan Chen, Mengyue Wu, Kenny Q. Zhu, Kunyao Lan, Zhiling Zhang, and Lyuchun Cui. *LLM-empowered Chatbots for Psychiatrist and Patient Simulation: Application and Evaluation*. 2023. arXiv: 2305.13614 [cs.CL] (cit. on p. 8).

[23] Junjie Ye et al. *A Comprehensive Capability Analysis of GPT-3 and GPT-3.5 Series Models*. 2023. arXiv: 2303.10420 [cs.CL] (cit. on p. 8).

[24] Christian Janiesch, Patrick Zschech, and Kai Heinrich. «Machine learning and deep learning». In: *Electronic Markets* 31.3 (Apr. 2021), pp. 685–695. ISSN: 1422-8890. DOI: 10.1007/s12525-021-00475-2. URL: http://dx.doi.org/10.1007/s12525-021-00475-2 (cit. on p. 8).

[25] Jianyi Zhang, Xu Ji, Zhangchi Zhao, Xiali Hei, and Kim-Kwang Raymond Choo. *Ethical Considerations and Policy Implications for Large Language Models: Guiding Responsible Development and Deployment*. 2023. DOI: 10.48550/ARXIV.2308.02678. URL: https://arxiv.org/abs/2308.02678 (cit. on p. 8).

[26] N. Gowri Vidhya, D. Devi, Nithya A., and T. Manju. «Prognosis of exploration on Chat GPT with artificial intelligence ethics». In: *Brazilian Journal of Science* 2.9 (Apr. 2023), pp. 60–69. ISSN: 2764-3417. DOI: 10.14295/bjs.v2i9.372. URL: http://dx.doi.org/10.14295/bjs.v2i9.372 (cit. on p. 8).

[27] Lixiang Yan, Lele Sha, Linxuan Zhao, Yuheng Li, Roberto Martinez-Maldonado, Guanliang Chen, Xinyu Li, Yueqiao Jin, and Dragan Gašević. «Practical and Ethical Challenges of Large Language Models in Education: A Systematic Scoping Review». In: *British Journal of Educational Technology* n/a.n/a (). ISSN: 1467-8535. DOI: 10.1111/bjet.13370 (cit. on p. 8).

[28] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. *A Comprehensive Overview of Large Language Models*. Dec. 2023. DOI: 10.48550/arXiv.2307.06435. arXiv: 2307.06435 [cs] (cit. on p. 8).

[29] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. *Exploring the Limits of Language Modeling*. Feb. 2016. DOI: 10.48550/arXiv.1602.02410. arXiv: 1602.02410 [cs] (cit. on p. 8).

[30] Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. *Editing Large Language Models: Problems, Methods, and Opportunities*. Nov. 2023. DOI: 10.48550/arXiv.2305.13172. arXiv: 2305.13172 [cs] (cit. on p. 8).

[31] Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. *Challenges and Applications of Large Language Models*. July 2023. DOI: 10.48550/arXiv.2307.10169. arXiv: 2307.10169 [cs] (cit. on p. 9).

[32] Wayne Xin Zhao et al. *A Survey of Large Language Models*. Nov. 2023. DOI: 10.48550/arXiv.2303.18223. arXiv: 2303.18223 [cs] (cit. on pp. 9, 14).

[33] Kai He, Rui Mao, Qika Lin, Yucheng Ruan, Xiang Lan, Mengling Feng, and Erik Cambria. *A Survey of Large Language Models for Healthcare: From Data, Technology, and Applications to Accountability and Ethics*. Oct. 2023. arXiv: 2310.05694 [cs] (cit. on p. 9).

[34] Piotr Mirowski, Kory W. Mathewson, Jaylen Pittman, and Richard Evans. *Co-Writing Screenplays and Theatre Scripts with Language Models: An Evaluation by Industry Professionals*. Sept. 2022. DOI: 10.48550/arXiv.2209.14958. arXiv: 2209.14958 [cs] (cit. on pp. 9, 10, 34).

[35] Rudolf Rosa, Patrícia Schmidtová, Ondřej Dušek, Tomáš Musil, David Mareček, Saad Obaid, Marie Nováková, Klára Vosecká, and Josef Doležal. «GPT-2-based Human-in-the-loop Theatre Play Script Generation». In: *Proceedings of the 4th Workshop of Narrative Understanding (WNU2022)*. Ed. by Elizabeth Clark, Faeze Brahman, and Mohit Iyyer. Seattle, United States: Association for Computational Linguistics, July 2022, pp. 29–37. DOI: 10.18653/v1/2022.wnu-1.4. URL: https://aclanthology.org/2022.wnu-1.4 (cit. on p. 9).

[36] Juntae Kim, Yoonseok Heo, Hogeon Yu, and Jongho Nang. «A Multi-Modal Story Generation Framework with AI-Driven Storyline Guidance». In: *Electronics* 12.6 (Mar. 2023), p. 1289. ISSN: 2079-9292. DOI: 10.3390/electronics12061289. URL: http://dx.doi.org/10.3390/electronics12061289 (cit. on p. 9).

[37] Shyam Sudhakaran, Miguel González-Duque, Claire Glanois, Matthias Freiberger, Elias Najarro, and Sebastian Risi. *MarioGPT: Open-Ended Text2Level Generation through Large Language Models*. June 2023. DOI: 10.48550/arXiv.2302.05981. arXiv: 2302.05981 [cs] (cit. on pp. 9, 59).

[38] Zhe Hu, Hou Pong Chan, Jiachen Liu, Xinyan Xiao, Hua Wu, and Lifu Huang. *PLANET: Dynamic Content Planning in Autoregressive Transformers for Long-form Text Generation*. 2022. arXiv: 2203.09100 [cs.CL] (cit. on p. 9).

[39] Juntae Kim, Yoonseok Heo, Hogeon Yu, and Jongho Nang. «A Multi-Modal Story Generation Framework with AI-Driven Storyline Guidance». In: *Electronics* 12.6 (2023). ISSN: 2079-9292. DOI: 10.3390/electronics12061289. URL: https://www.mdpi.com/2079-9292/12/6/1289 (cit. on p. 9).

[40] Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. «Table-to-Text Generation by Structure-Aware Seq2seq Learning». In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (Apr. 2018). ISSN: 2159-5399. DOI: 10.1609/aaai.v32i1.11925. URL: http://dx.doi.org/10.1609/aaai.v32i1.11925 (cit. on p. 9).

[41] Shuming Ma, Pengcheng Yang, Tianyu Liu, Peng Li, Jie Zhou, and Xu Sun. «Key Fact as Pivot: A Two-Stage Model for Low Resource Table-to-Text Generation». In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Ed. by Anna Korhonen, David Traum, and Lluís Màrquez. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 2047–2057. DOI: 10.18653/v1/P19-1197. URL: https://aclanthology.org/P19-1197 (cit. on p. 9).

[42] Deepak Nanuru Yagamurthy, Rajesh Azmeera, and Rhea Khanna. «Natural Language Generation (NLG) for Automated Report Generation». In: *Journal of Technology and Systems* 5.1 (Nov. 2023), pp. 48–59. DOI: 10.47941/jts.1497. URL: https://carijournals.org/journals/index.php/JTS/article/view/1497 (cit. on p. 9).

[43] Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. «Few-Shot NLG with Pre-Trained Language Model». In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault. Online: Association for Computational Linguistics, July 2020, pp. 183–190. DOI: 10.18653/v1/2020.acl-main.18. URL: https://aclanthology.org/2020.acl-main.18 (cit. on p. 9).

[44] Yulun Du and Lydia Chilton. *STORYWARS: A Dataset and Instruction Tuning Baselines for Collaborative Story Understanding and Generation*. 2023. arXiv: 2305.08152 [cs.CL] (cit. on p. 9).

[45] Tae Soo Kim, Yoonjoo Lee, Jamin Shin, Young-Ho Kim, and Juho Kim. *EvalLM: Interactive Evaluation of Large Language Model Prompts on User-Defined Criteria*. 2023. DOI: 10.48550/ARXIV.2309.13633. URL: https://arxiv.org/abs/2309.13633 (cit. on pp. 10, 11).

[46] Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. «Generative Judge for Evaluating Alignment». In: *ArXiv* abs/2310.05470 (2023). URL: https://api.semanticscholar.org/CorpusID:263829791 (cit. on p. 10).

[47] Long Ouyang et al. *Training language models to follow instructions with human feedback*. 2022. arXiv: 2203.02155 [cs.CL] (cit. on pp. 10, 31).

[48] Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. *Preference Ranking Optimization for Human Alignment*. 2023. arXiv: 2306.17492 [cs.CL] (cit. on p. 10).

[49] Zhiqing Sun, Yikang Shen, Hongxin Zhang, Qinhong Zhou, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. *SALMON: Self-Alignment with Principle-Following Reward Models*. 2023. arXiv: 2310.05910 [cs.CL] (cit. on p. 10).

[50] Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. *Black-Box Prompt Optimization: Aligning Large Language Models without Model Training*. 2023. arXiv: 2311.04155 [cs.CL] (cit. on p. 10).

[51] Haoran Li, Yiran Liu, Xingxing Zhang, Wei Lu, and Furu Wei. *Tuna: Instruction Tuning using Feedback from Large Language Models*. 2023. arXiv: 2310.13385 [cs.CL] (cit. on p. 10).

[52] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. *G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment*. 2023. arXiv: 2303.16634 [cs.CL] (cit. on p. 10).

[53] Mingqi Gao, Xinyu Hu, Jie Ruan, Xiao Pu, and Xiaojun Wan. *LLM-based NLG Evaluation: Current Status and Challenges*. 2024. arXiv: 2402.01383 [cs.CL] (cit. on p. 10).

[54] Cheng-Han Chiang and Hung-yi Lee. *Can Large Language Models Be an Alternative to Human Evaluations?* 2023. arXiv: 2305.01937 [cs.CL] (cit. on p. 11).

[55] Tom Kocmi and Christian Federmann. *Large Language Models Are State-of-the-Art Evaluators of Translation Quality*. 2023. arXiv: 2302.14520 [cs.CL] (cit. on p. 11).

[56] Anja Belz and Ehud Reiter. «Comparing automatic and human evaluation of NLG systems». In: *11th conference of the european chapter of the association for computational linguistics*. 2006, pp. 313–320 (cit. on p. 11).

[57] Zheheng Luo, Qianqian Xie, and Sophia Ananiadou. *ChatGPT as a Factual Inconsistency Evaluator for Text Summarization*. Apr. 2023. arXiv: 2303.15621 [cs] (cit. on p. 11).

[58] Adian Liusie, Potsawee Manakul, and Mark J. F. Gales. *LLM Comparative Assessment: Zero-shot NLG Evaluation through Pairwise Comparisons using Large Language Models*. 2024. arXiv: 2307.07889 [cs.CL] (cit. on p. 11).

[59] Yebowen Hu, Kaiqiang Song, Sangwoo Cho, Xiaoyang Wang, Hassan Foroosh, and Fei Liu. «DecipherPref: Analyzing Influential Factors in Human Preference Judgments via GPT-4». In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 8344–8357. DOI: 10.18653/v1/2023.emnlp-main.519. URL: https://aclanthology.org/2023.emnlp-main.519 (cit. on p. 11).

[60] Mingqi Gao, Jie Ruan, Renliang Sun, Xunjian Yin, Shiping Yang, and Xiaojun Wan. *Human-like Summarization Evaluation with ChatGPT*. 2023. arXiv: 2304.02554 [cs.CL] (cit. on p. 11).

[61] Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. *SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models*. 2023. arXiv: 2303.08896 [cs.CL] (cit. on p. 11).

[62] Yuxuan Liu, Tianchi Yang, Shaohan Huang, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. *Calibrating LLM-Based Evaluator*. 2023. arXiv: 2309.13308 [cs.CL] (cit. on p. 11).

[63] Friedrich Steinhäusler. «EU Including: Development of Radiological and Nuclear Training Learning Objectives». In: *Journal of Applied Mathematics and Physics* 09.08 (2021), pp. 2170–2178. ISSN: 2327-4379. DOI: 10.4236/jamp.2021.98136. URL: http://dx.doi.org/10.4236/jamp.2021.98136 (cit. on p. 12).

[64] L van Puyvelde, T Clarijs, N Belmans, and M Coeck. «Comparing the effectiveness of learning formats in radiation protection». In: *Journal of Radiological Protection* 41.4 (Sept. 2021), pp. 707–725. ISSN: 1361-6498. DOI: 10.1088/1361-6498/ac0803. URL: http://dx.doi.org/10.1088/1361-6498/ac0803 (cit. on p. 12).

[65] Zonghan Yang, Xiaoyuan Yi, Peng Li, Yang Liu, and Xing Xie. *Unified Detoxifying and Debiasing in Language Generation via Inference-time Adaptive Optimization*. 2023. arXiv: 2210.04492 [cs.CL] (cit. on p. 12).

[66] Laura Weidinger et al. «Ethical and social risks of harm from Language Models». In: *ArXiv* abs/2112.04359 (2021). URL: https://api.semanticscholar.org/CorpusID:244954639 (cit. on p. 12).

[67] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Eric Sun, and Yue Zhang. «A survey on large language model (llm) security and privacy: The good, the bad, and the ugly». In: *arXiv preprint arXiv:2312.02003* (2023) (cit. on p. 12).

[68] Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. «Privacy Risks of General-Purpose Language Models». In: *2020 IEEE Symposium on Security and Privacy (SP)*. 2020, pp. 1314–1331. DOI: 10.1109/SP40000.2020.00095 (cit. on p. 12).

[69] Arshiya Aggarwal, Jiao Sun, and Nanyun Peng. *Towards Robust NLG Bias Evaluation with Syntactically-diverse Prompts*. 2022. arXiv: 2212.01700 [cs.CL] (cit. on p. 12).

[70] Kaitlyn Zhou, Su Lin Blodgett, Adam Trischler, Hal Daumé III au2, Kaheer Suleman, and Alexandra Olteanu. *Deconstructing NLG Evaluation: Evaluation Practices, Assumptions, and Their Implications*. 2022. arXiv: 2205.06828 [cs.CL] (cit. on p. 12).

[71] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. «State-of-the-Art in Artificial Neural Network Applications: A Survey». In: *Heliyon* 4.11 (Nov. 2018), e00938. ISSN: 24058440. DOI: 10.1016/j.heliyon.2018.e00938 (cit. on p. 16).

[72] F Rosenblatt. «The perceptron: a probabilistic model for information storage and organization in the brain». en. In: *Psychol. Rev.* 65.6 (Nov. 1958), pp. 386–408 (cit. on p. 16).

[73] Sepp Hochreiter and Jürgen Schmidhuber. «Long short-term memory». In: *Neural computation* 9.8 (1997), pp. 1735–1780. URL: http://www.bioinf.jku.at/publications/older/2604.pdf (cit. on p. 20).

[74] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. «Dropout: A Simple Way to Prevent Neural Networks from Overfitting». In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html (cit. on p. 23).

[75] Maksym Andriushchenko, Francesco D'Angelo, Aditya Varre, and Nicolas Flammarion. *Why Do We Need Weight Decay in Modern Deep Learning?* 2023. arXiv: 2310.04415 [cs.LG] (cit. on p. 23).

[76] Igor Grlicarev - Slovenian Nuclear Safety Administration. *EFFECTIVE NUCLEAR AND RADIATION EMERGENCY PLANNING*. https://inis.iaea.org/collection/NCLCollectionStore/_Public/31/051/31051416.pdf (cit. on p. 29).

[77] IAEA. *RADIOLOGICAL EMERGENCY AT RESEARCH REACTOR*. https://www-pub.iaea.org/MTCD/Publications/PDF/SupplementaryMaterials/EPR-Research_Reactor_CD/TabletopExerciseManualSection1.pdf (cit. on p. 29).

[78] IAEA. *Radiological Emergency Response Exercise - Medical scenario - Exercise Manual*. https://www-pub.iaea.org/mtcd/publications/pdf/eprexert/PDF/MedExManual.pdf. 2005 (cit. on p. 29).

[79] IAEA. *Tabletop Exercise Scenario – Basic - Assessment of Patient's Exposure*. https://www-pub.iaea.org/MTCD/publications/PDF/EPR-Medical_T_2011/PDF/Tabletop_Exercise.pdf (cit. on p. 29).

[80] EU's 7th Framework Programme. *D2.2 REFERENCE SET OF CBRN SCENARIOS*. https://ffi-publikasjoner.archive.knowledgearc.net/bitstream/handle/20.500.12242/1613/D2-2-Reference-set-of-CBRN-scenarios.pdf (cit. on p. 29).

[81] OECD. *Strategy for Developing and Conducting Nuclear Emergency Exercises - OECD*. https://www.oecd-nea.org/upload/docs/application/pdf/2019-12/6162-emergency.pdf. 2007 (cit. on p. 29).

[82] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. *A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT*. 2023. arXiv: 2302.11382 [cs.SE] (cit. on p. 31).

[83] Rick Battle and Teja Gollapudi. *The Unreasonable Effectiveness of Eccentric Automatic Prompts*. 2024. arXiv: 2402.10949 [cs.CL] (cit. on p. 31).

[84] Lei Wang, Wenshuai Bi, Suling Zhao, Yinyao Ma, Longting Lv, Chenwei Meng, Jingru Fu, and Hanlin Lv. «Investigating the Impact of Prompt Engineering on the Performance of Large Language Models for Standardizing Obstetric Diagnosis Text: Comparative Study». In: *JMIR Formative Research* 8 (Feb. 2024), e53216. ISSN: 2561-326X. DOI: 10.2196/53216. URL: http://dx.doi.org/10.2196/53216 (cit. on p. 31).

[85] Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, and Yang Liu. *Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study*. 2023. arXiv: 2305.13860 [cs.SE] (cit. on p. 31).

[86] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: 2201.11903 [cs.CL] (cit. on p. 33).

[87] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. *Large Language Models are Zero-Shot Reasoners*. 2023. arXiv: 2205.11916 [cs.CL] (cit. on p. 33).

[88] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. *Tree of Thoughts: Deliberate Problem Solving with Large Language Models*. 2023. arXiv: 2305.10601 [cs.CL] (cit. on p. 33).

[89] Dave Hulbert. *Using Tree-of-Thought Prompting to boost ChatGPT's reasoning*. https://github.com/dave1010/tree-of-thought-prompting. May 2023. DOI: 10.5281/ZENODO.10323452. URL: https://doi.org/10.5281/zenodo.10323452 (cit. on pp. 33, 35).

[90] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL] (cit. on p. 33).

[91] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. *Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?* 2022. arXiv: 2202.12837 [cs.CL] (cit. on p. 33).

[92] Yupeng Chang et al. *A Survey on Evaluation of Large Language Models*. 2023. arXiv: 2307.03109 [cs.CL] (cit. on p. 37).

[93] Debarag Banerjee, Pooja Singh, Arjun Avadhanam, and Saksham Srivastava. *Benchmarking LLM powered Chatbots: Methods and Metrics*. 2023. arXiv: 2308.04624 [cs.CL] (cit. on p. 38).

[94] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. *Just Ask for Calibration: Strategies for Eliciting Calibrated Confidence Scores from Language Models Fine-Tuned with Human Feedback*. 2023. arXiv: 2305.14975 [cs.CL] (cit. on p. 38).

[95] Boxin Wang et al. *DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models*. 2024. arXiv: 2306.11698 [cs.CL] (cit. on p. 38).

[96] Jindong Wang et al. *On the Robustness of ChatGPT: An Adversarial and Out-of-distribution Perspective*. 2023. arXiv: 2302.12095 [cs.AI] (cit. on p. 38).

[97] Kaijie Zhu et al. *PromptBench: Towards Evaluating the Robustness of Large Language Models on Adversarial Prompts*. 2023. arXiv: 2306.04528 [cs.CL] (cit. on p. 38).

[98] FEMA. *Developing and Maintaining Emergency Operations Plans*. https://www.fema.gov/sites/default/files/documents/fema_cpg-101-v3-developing-maintaining-eops.pdf (cit. on p. 40).

[99] Yidong Wang et al. *PandaLM: An Automatic Evaluation Benchmark for LLM Instruction Tuning Optimization*. 2023. arXiv: 2306.05087 [cs.CL] (cit. on p. 40).

[100] Swarnadeep Saha, Omer Levy, Asli Celikyilmaz, Mohit Bansal, Jason Weston, and Xian Li. *Branch-Solve-Merge Improves Large Language Model Evaluation and Generation*. 2023. arXiv: 2310.15123 [cs.CL] (cit. on p. 40).

[101] Sue Lim and Ralf Schmälzle. «Artificial intelligence for health message generation: an empirical study using a large language model (LLM) and prompt engineering». In: *Frontiers in Communication* 8 (May 2023). ISSN: 2297-900X. DOI: 10.3389/fcomm.2023.1129082. URL: http://dx.doi.org/10.3389/fcomm.2023.1129082 (cit. on p. 40).

[102] Xingdi Yuan, Tong Wang, Yen-Hsiang Wang, Emery Fine, Rania Abdelghani, Pauline Lucas, Hélène Sauzéon, and Pierre-Yves Oudeyer. *Selecting Better Samples from Pre-trained LLMs: A Case Study on Question Generation*. 2022. arXiv: 2209.11000 [cs.CL] (cit. on p. 40).

[103] Shiran Dudy, Steven Bedrick, and Bonnie Webber. *Refocusing on Relevance: Personalization in NLG*. 2021. arXiv: 2109.05140 [cs.CL] (cit. on p. 40).

[104] Yasuhide Miura, Yuhao Zhang, Emily Bao Tsai, Curtis P. Langlotz, and Dan Jurafsky. *Improving Factual Completeness and Consistency of Image-to-Text Radiology Report Generation*. 2021. arXiv: 2010.10042 [cs.CL] (cit. on p. 40).

[105] Xuanyu Zhang, Bingbing Li, and Qing Yang. *CGCE: A Chinese Generative Chat Evaluation Benchmark for General and Financial Domains*. 2023. arXiv: 2305.14471 [cs.CL] (cit. on p. 40).

[106] Chujie Gao, Dongping Chen, Qihui Zhang, Yue Huang, Yao Wan, and Lichao Sun. *LLM-as-a-Coauthor: The Challenges of Detecting LLM-Human Mixcase*. 2024. arXiv: 2401.05952 [cs.CL] (cit. on p. 40).

[107] Isabelle Augenstein et al. *Factuality Challenges in the Era of Large Language Models*. 2023. arXiv: 2310.05189 [cs.CL] (cit. on p. 41).

[108] Yejin Bang et al. *A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity*. 2023. arXiv: 2302.04023 [cs.CL] (cit. on p. 41).

[109] Denis Peskoff and Brandon Stewart. «Credible without Credit: Domain Experts Assess Generative Language Models». In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Ed. by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 427–438. DOI: 10.18653/v1/2023.acl-short.37. URL: https://aclanthology.org/2023.acl-short.37 (cit. on p. 41).

[110] Stuart E. Dreyfus. «The Five-Stage Model of Adult Skill Acquisition». In: *Bulletin of Science, Technology & Society* 24.3 (2004), pp. 177–181. DOI: 10.1177/0270467604264992. eprint: https://doi.org/10.1177/0270467604264992. URL: https://doi.org/10.1177/0270467604264992 (cit. on p. 41).

[111] National Statistician's Data Ethics Advisory Committee. *Guidelines on using the ethics self-assessment process*. https://uksa.statisticsauthority.gov.uk/wp-content/uploads/2021/04/2021_Self-assessment_guidance_V2.3.pdf (cit. on p. 51).