JASSS
THE JOURNAL OF ARTIFICIAL SOCIETIES AND SOCIAL SIMULATION

From Concepts to Model: Automating Feature Extraction of Agent-based Models using Large Language Models

Siamak Khatami[1] **and Christopher Frantz**[1]

[1]*Department of Computer Science, Norwegian University of Science and Technology, Teknologivegen 22, Gjøvik, 2815, Norway*
Correspondence should be addressed to *siamak.khatami@ntnu.no*

Journal of Artificial Societies and Social Simulation 28(3) 9, 2025
Doi: 10.18564/jasss.5734 Url: http://jasss.soc.surrey.ac.uk/28/3/9.html

Received: 28-08-2024 Accepted: 10-06-2025 Published: 30-06-2025

Abstract: Converting conceptual models into simulation models is a primary challenge in the Agent-based Modeling (ABM) development lifecycle, often acting as a bottleneck due to communication gaps among different stakeholders, particularly programmers and mathematicians. To address this issue, LargeLanguageModels (LLMs) and, more specifically, Question-answering (QA) models (also known as Conversational Artificial Intelligence (CAI)) can play a central role. However, using QA models and related Natural Language Processing (NLP) techniques for auto-generating simulation models in an integrated process is promising with respect to increasing efficiency, consistency and potentially quality of the extracted conceptual model. Drawing on contemporary QA models, our proposed approach involves the systematic extraction of model features from descriptions to generate a conceptual model that is the precursor for automating the generation of model implementations. A central contribution of this work is to establish a systematic method for this initial step, extracting simulation-relevant information from the conceptual model and presenting it in a generic machine- and human-readable format suitable for future applications facilitating automated code generation, taking into account the continuous technological progression in this area. To this end, this article introduces a baseline schema of information pertinent to developing basic conceptual ABMs and employs a testbed methodology to examine the performance of a wide range of contemporary LLMs like Llama 2 and 3 embedded in QA models, alongside available commercial QAs (like ChatGPT-3.5 and ChatGPT-4o). The evaluation relies on a combination of automated cosine similarity and manual similarity assessment to establish generic metrics to assess QA model performance. The results of contemporary models show the dominant performance of commercially available models (ChatGPT-4o), while highlighting the increasing potential for the use of open source LLMs (e.g., Llama 2 and 3) for the purpose of model feature extraction. We discuss the associated implications for automated code generation as well as future directions while also exploring the broader potential for the use of QA models to support other stages of the development cycle of ABMs.

Keywords: Model Extraction, Artificial Intelligence, Natural Language Processing, Large Language Models, Question Answering, Agent-based Modeling

Introduction

1.1 ABM enables individuals and organizations to study complex social systems from a bottom-up perspective by modeling individual agents and observing their concerted behavior by drawing on a bottom-up metaphor. The interactions between these individual agents, along with their randomness and personalized behaviors based on individual-level heterogeneity, make the aggregated and macro-level behaviors unpredictable in real life, reflecting its potential to mirror the complexity found in human systems. Despite its development challenges (which will be discussed later), ABM is a suitable method for policymakers and researchers to study social phenomena (Edmonds & Meyer 2017).

1.2 ABM development is a multifaceted process that entails several critical steps and challenges. Broadly, it can be divided into three key phases: conceptualization, implementation, and evaluation and analysis. Numerous studies have explored various dimensions of these phases, including aspects such as conceptualization (Edmonds & Moss 2005; Howick et al. 2024), contextualization (Edmonds 2015a, 2012), simulation coding, verification, validation, and adherence to development protocols (Grimm et al. 2020b). Furthermore, a variety of development life cycles have been proposed (Edmonds & Meyer 2017; Gilbert & Troitzsch 2005).

1.3 Figure 1 presents a life cycle model for ABM development, drawing insights from these studies, particularly the Keep It Descriptive Stupid (KIDS) framework and the Overview, Design concepts, Details (ODD) protocol. This model emphasizes both the procedural (step-by-step) and analytical (qualitative and quantitative) approaches necessary for effective ABM development.

1.4 The roadmap depicted in Figure 1 outlines a 10-step ABM development life cycle that begins with observing and understanding the real world, followed by efforts to explain it through modeling. This framework provides a mid-level representation of the ABM development process, focusing on individual steps rather than broader categories like conceptualization, implementation, and evaluation by synthesising previous efforts in the respective areas. We hence intentionally focus on the key steps involved in the development process, but avoid delving into the finer details of each step.

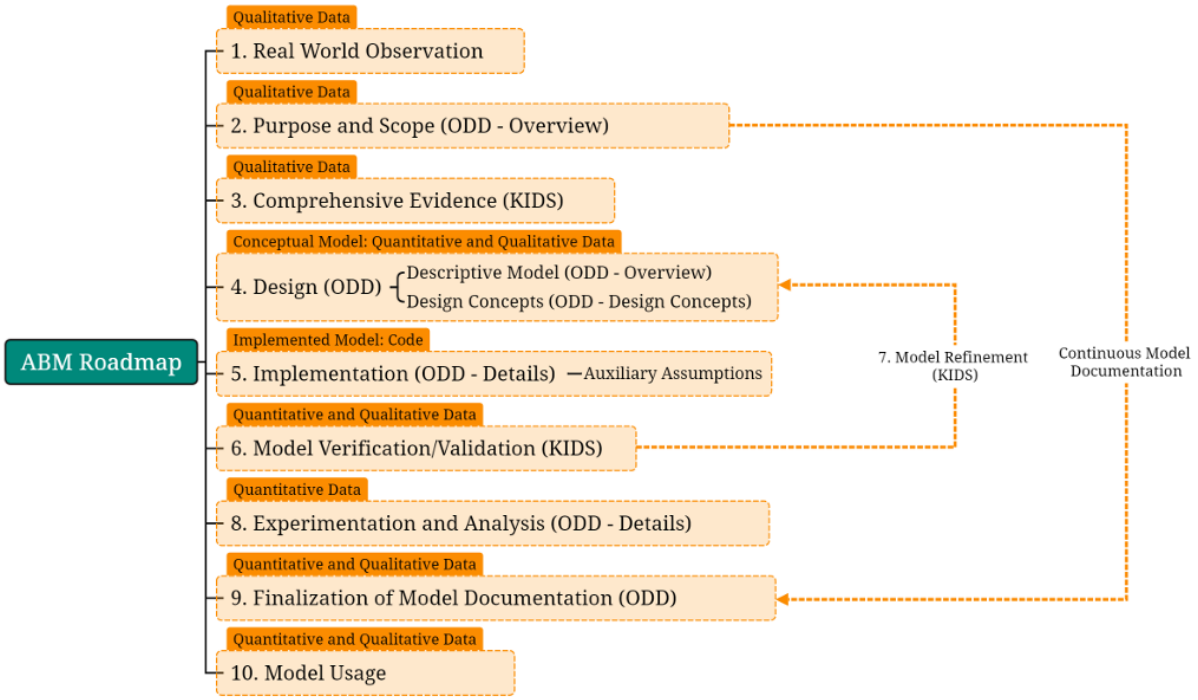

Figure 1: ABM Roadmap synthesised based on KIDS protocol and ODD standard (Edmonds & Moss 2005; Grimm et al. 2020b; Edmonds & Meyer 2017)

1.5 Central to this framework are the *KIDS* and *ODD* protocols, which play pivotal roles. The KIDS framework emphasizes the importance of incorporating all relevant details into the modeling process, ensuring the model captures the complexity of the real world. Meanwhile, the ODD protocol facilitates the abstraction, construction, and efficient documentation and presentation of the model. Together, these protocols balance detail-oriented rigor with the need for clarity and accessibility in ABM development.

1.6 The KIDS approach emphasizes the inclusion of detailed and comprehensive information in ABM development, as highlighted by Edmonds & Moss. This perspective stems from discussions within the computer-engineering domain of ABM development, where the modeling process traditionally starts with a simplified system and iteratively expands—a method known as *Keep It Simple Stupid (KISS)* (Edmonds & Moss 2005). However, the KIDS framework addresses a significant challenge: the collection, filtering, and processing of vast and complex narrative data, which is both time-intensive and knowledge-demanding. This challenge is a recognized pitfall

in social simulation, often identified as one of the primary hurdles in the field (Edmonds & Moss 2005; Edmonds 2015a,b).

1.7 The adoption of the KIDS approach in drawing the ABM life cycle, as depicted in Figure 1, is justified for several reasons. First, it aligns closely with the intrinsic nature and structure of complex phenomena. Second, it promotes a more effective utilization of available evidence. Third, it avoids unwarranted simplifications that could compromise the model's accuracy. Finally, it enhances the descriptive and empirical precision of the model, which in terms of complexity analysis is necessary (Sterman 2000). These attributes make KIDS particularly effective when paired with the *ODD* protocol (Grimm et al. 2006, 2010; Müller et al. 2013; Grimm et al. 2020b), which provides a robust framework and best practices for describing ABMs. Together, they establish a comprehensive and methodologically sound foundation for ABM development.

1.8 Each of ABM development steps consists of a number of challenges, preventing ABM from being used by potential stakeholders (Galán et al. 2009, 2013; Galan & Pavón 2009; An et al. 2020; Grimm et al. 2020a). Even though the nature of the process in each step is different and solutions that may be suggested should also differ in detail, these challenges can be divided into two main categories: qualitative and quantitative. Qualitative challenges include issues like conceptual model inconsistency (Edmonds & Moss 2005; Galán et al. 2009) due to its use of natural language, comprehensive analysis of narratives (Edmonds 2015a) as well as communication problems between different developer roles, such as thematician, computer scientists, and users (Galán et al. 2009). On the other hand, quantitative challenges involve behavior modeling in both conceptual and implemented versions (Galán et al. 2009; Khatami & Frantz 2023) as well as different behavior types from fully autonomous behavioral modeling till nonlinear mathematical abstraction of a behavior. However, one of the critical challenges is the interactive transmission from conceptual models to implemented simulation models. This difficulty arises from human shortcomings in analyzing large amounts of text, as well as variation in text and the emergence of communication between different role players (Galán et al. 2009; Edmonds 2015a, 2012; Edmonds & Moss 2005). Although the ABM process begins with the collection of narratives, our focus does not lie on the analysis of narrative inputs but rather on the conceptual document typically produced following the development of the conceptual model. The analysis of narratives is hence beyond the scope of this study. This decision is driven by the key challenges observed in modeling, which stem from the communication and time-consuming interaction between domain experts and computer scientists. Typically, one party articulates information in natural language while the other grapples with the practical and theoretical limitations (Galán et al. 2009). Recognizing this pattern, the communication and conceptual model improvement cycle can become more effective. Furthermore, the conceptual model is often confined by assumptions, whereas narratives can expand these assumptions by focusing on diverse scientific fields, new evidence, and observations, hence offering elaborations beyond the vague initial stance as well as unclear boundaries of the frame. By treating narrative analysis and conceptualization as exogenous topics but relevant inputs to this study, the focus shifts to examining how recent advances in computer science can facilitate the conversion of conceptual models into implemented models.

1.9 The rise of computational techniques, particularly in Deep Learning (DL) and NLP, offers promising solutions for addressing the challenges of analyzing large textual datasets. This study proposes a two-step method: first, extracting information from conceptual models using QA models, which are critical for simulation model generation; and second, employing auto-code generation to build the model from the extracted information. The focus here in this study is on the first step – information extraction – while discussing the rationale behind this method selection as well as presenting the information in a schema to be used in all kinds of auto-code generation methods, either QA or DL models.

1.10 NLP is a branch of deep learning that focuses on processing human language for various tasks such as text summarization, classification, clustering, sentiment analysis, and QA (Khurana et al. 2023; Lan et al. 2021). The introduction of the attention mechanism and transformers in NLP (Vaswani et al. 2017) has led to the development of pre-trained LLMs that can be trained over long texts, which allow us to process large bodies of text and perform different tasks like question answering, information extraction and information summarization. However, the variety of NLP techniques varies, and many different architectures can be provided for summarizing and extracting information to focus on the conceptual-to-code challenge. Given the long-term aspiration to develop NLP's usefulness to advance the development of ABMs from narrative inputs to automated code generation, there can be two different solution pathways.

1.11 The first and strongly technical solution is a fully black-box DL architecture that retrieves the text and generates executable simulation code (Padilla et al. 2019; Paudel & Ligmann-Zielinska 2023). While advancements in NLP and LLMs have sparked new directions for streamlining and automating ABM development, it is important to note that simply applying NLP techniques to automate the ABM process is not a universal solution. By their very nature as a semi-formal approach (unlike game-theoretical models, for example), the theoretical

and narrative approaches used to capture ABMs commonly introduce gaps, errors, and artefacts. While efficient coding and addressing narrative issues can help address selected ABM challenges, it does not guarantee a comprehensive solution due to the intransparent black-box nature of DL architectures. Although efficient coding and addressing narrative inconsistencies can mitigate some ABM challenges, the inherent black-box nature of DL architectures limits their transparency and interpretability, rendering them as a suboptimal for capturing model information comprehensively. Moreover, the prospect of generating code from complex and lengthy documents remains challenging yet promising. While automatically generated models demonstrate impressive performance for well-specified small-scale cases, creating a fully operational application based solely on a descriptive input and establishing seamless linkages between model features and functionality has yet to be shown as a reliable solution (Liu et al. 2024, 2023b).

1.12 The second approach proposes a two-step method: (1) extracting and validating the simulation-required information with iterative conceptual correction and (2) auto-generating code using LLMs. This study focuses on the first step—extracting information from conceptual models using QA techniques and structuring it in formats that are both human- and machine-readable, such as JavaScript Object Notation (JSON) or Extensible Markup Language (XML).

1.13 Given the complexities of auto-code generation (Liu et al. 2024, 2023b), the initial emphasis is placed on summarizing the central conceptual features of an ABM into a structured format. This approach not only allows for validating the accuracy and completeness of the extracted information but also ensures flexibility for exploring diverse strategies in downstream code generation. As a result, this study concentrates on step 1, encompassing both information extraction and validation, while deferring the evaluation of auto-generated simulation code to future work. By addressing the foundational aspects systematically, this study aims to establish a reliable framework for advancing ABM automation in a longer aim.

1.14 LLM-based general purpose techniques like QA – also known as conversational Artificial Intelligence (AI) can be used more straightforward. QA is a branch of NLP in which a pre-trained LLM functions as the core of the model while providing context information (e.g., in the form of documents). These models are designed to calculate the possible likelihood of the arrangement of words based on the input conversion (also known as prompt, task, and input text) and the text that they have been trained on. Thus, the core LLM in these models and the amount of text they have been trained with matters, which will be discussed in Section 3. The main reason for exploring this approach lies in the increasingly accessible use of QA based on the continuous development in the area of computer science, which requires strong equipment, and massive data collection, offering challenges that are outside the simulation community's main focus. Recognizing that the simulation community will benefit from the continued rapid development of LLM models, we embed QA models as part of the methodology that we use in this study. Instead of drawing on a particular model, we will hence devise a benchmark prompt system to extract relevant information from conceptual models using QA models (as detailed in Section 3), with the objective to represent this information in intermediate data structures based on open formats such as JSON, which can then be utilized by parallel scientific efforts that focus on automated code generation of simulation models (Vaithilingam et al. 2022; Liu et al. 2023a). To achieve this, we will use a selection of QA models and a set of ABM models as the input from which the information will be retrieved.

1.15 The decision to summarize extracted information into machine-readable schemas like JSON rather than directly using the ODD protocol stems from two primary reasons. First, formats like JSON are inherently machine-readable, comprising key-value pairs, unlike ODD, which is less suitable for direct computational processing. Second, many auto-code generator models accept inputs in formats such as JSON, UML, or XML and cannot process ODD text structures. While it is feasible to provide ODD as a context to QA models for direct code generation, the current limitations of QA models in handling complex systems for auto-code generation (Chen & Wilensky 2023; Liu et al. 2024, 2023b) require more advanced LLMs and specialized research (Siebers 2025; Chen & Wilensky 2023). Third, JSON has become the contemporary de-facto standard for API development, including ChatGPT.

1.16 Contemporarily, there has been limited research on integrating LLMs into the ABM development life cycle. By addressing these gaps, this study aims to contribute to the systematic use of LLMs for advancing ABM development, focusing on extracting and structuring conceptual information while laying the groundwork for future automated code generation efforts.

Literature

1.17 Even though AI techniques have been used for the last few years, the invention of ChatGPT families was a revolution in the use of LLMs (Siebers 2025; Khatami & Frantz 2024c). This invention almost caught the scientific

field surprise. However, many studies have been started to be shaped around the usage of LLMs and CAI in ABM development lifecycle from different points of view.

1.18 Chen & Wilensky (2023) introduced ChatLogo, a hybrid interface that combines natural language with programming language to facilitate ABM development. Leveraging LLMs, ChatLogo enables users to interact with the modeling environment using conversational language, streamlining the coding process and making it more accessible to individuals without extensive programming expertise. The efficiency of this integration yet requires further investigations (Chen & Wilensky 2023). In a related study, Chen et al. (2024) examined the integration of LLM companions, such as ChatGPT, into the ABM learning process. Their findings suggest that LLM companions can assist both novices and experts in understanding and developing agent-based models by providing real-time feedback, explanations, and coding assistance, thereby enhancing the overall learning and development experience (Chen et al. 2024).

1.19 In addition, there are some studies relevant to filling the conceptualization gaps using LLMs. Siebers (2025) explored the potential of CAI systems like ChatGPT to support ABM model design. By employing advanced prompt engineering techniques and adhering to the engineering Agent-based Social Simulation (ABSS) framework, the study demonstrated how CAI can facilitate the development of innovative conceptual ABMs in a concise timeframe and with minimal required upfront case-based knowledge. Gurcan (2024) investigated the augmentation of agent-based simulations through the integration of LLMs, particularly for social simulations. The paper outlined architectures and methods to systematically develop LLM-augmented social simulations, discussing potential research directions in this field. Finally, they concluded that integrating LLMs with Agent-based Simulation (ABS) offers a powerful toolset for researchers, allowing for more nuanced, realistic, and comprehensive models of complex systems and human behaviors.

1.20 In another study, Shuttleworth & Padilla (2022) investigated the use of NLP techniques to semi-automatically generate conceptual models from textual descriptions of phenomena. By transforming narratives into lists of concepts and relationships, visualized through network graphs, the study demonstrates how pattern-based grammatical rules and NLP pipelines can streamline the initial stages of ABM development, reducing the manual effort required to create accurate conceptual models. Another study by Giabbanelli (2023), examines the application of LLMs like ChatGPT-4o in scientific simulations. Focusing on four modeling and simulation tasks, the authors assess the capabilities of LLMs in generating simulation code, interpreting simulation results, and automating aspects of the simulation process. The findings suggest that LLMs can significantly enhance efficiency and accuracy in ABM development by automating complex tasks traditionally performed by human modelers.

1.21 Padilla et al. (2019) presented a methodology that leverages NLP techniques to extract qualitative data from textual sources. Their approach aimed to facilitate the development of empirical ABMs by automating the extraction of relevant information, thereby reducing the manual effort required in the initial stages of model development. Another study, by Paudel & Ligmann-Zielinska (2023) examines the application of NLP methods to analyze and characterize existing ABM. By processing the documentation and descriptions of ABMs, they demonstrate how NLP can be used to identify model components, behaviors, and interactions, providing insights that can inform model refinement and development. The outputs of these studies show that relying on the complex architecture of DL and NLP in order to have a one-shot solution from conceptual models toward implemented models is not an efficient method at the moment. In summary, while all these studies aim to evaluate different usages of LLMs regarding ABM development, still the use of QA models in order to extract simulation information demands a systematic approach to be able to iteratively evaluate the new developments in NLP and especially LLM field.

1.22 Following our discussion about the nature of challenges, specification extraction can be categorized in the qualitative category. Recognizing the recent developments in the area of *NLP* based on attention mechanism and pre-trained LLMs (Petroni et al. 2019; Khurana et al. 2023; Padilla et al. 2019), we see promising opportunities in addressing some of the challenges by harmonizing the approaches for text extraction, but most importantly, to establish a descriptive baseline to explore opportunities and limitations, and to provide a basis for further refinement of QA models in filling the gaps relevant to large language documents processing and information extraction for different purposes in the ABM development life cycle in which information extraction from conceptual models is one of those challenges.

1.23 We outline the approach by initially examining the key elements to be extracted from conceptual models (Section 2), before exploring the methods for extracting and summarizing of information from documents using LLM in Section 3. Following this, in the Methodology section (Section 3), we will propose a systematic benchmark method, which begins by examining the key elements to be extracted from conceptual models (Section 2), before turning to the specification of a set of prompts (input text for question-answering models). The full

list of prompts relevant to the target information can be found in the separate document presented on arXiv (Khatami & Frantz 2024b).

1.24 Building on this, a set of ABMs will be selected as the context of the QA models. 'Context' in a QA model refers to a specific document that is given to the model, and prompts will be answered based on those documents. We will then showcase a series of experiments (Section 4) using LLM and ABM models and provide a thorough analysis of the reasons behind choosing these models. Concluding this paper, we will present and interpret the results (against an ideal baseline), and discuss their implications for the use of LLMs for the purpose of model information extraction as a preparation for automated code generation.

Methodology

2.1 As discussed in the introduction, the aim is to extract information from conceptual models. To do so, the first step is to discuss what is needed to be extracted. Afterward, we will explain our benchmark, which has been designed due to the absence of any other previous systematic work on this topic. The benchmark incorporates information about the ABMs that have been selected as part of the context in this study, QA models, prompts relevant to the target information, and validation of outputs. Following this setup, results will be illustrated and discussed.

Target Information

2.2 To develop an ABM, it is crucial to gather comprehensive information on the key features of ABMs and the contextual information essential for their successful implementation. Figure 2 provides an overview of a minimum set of relevant information pieces.

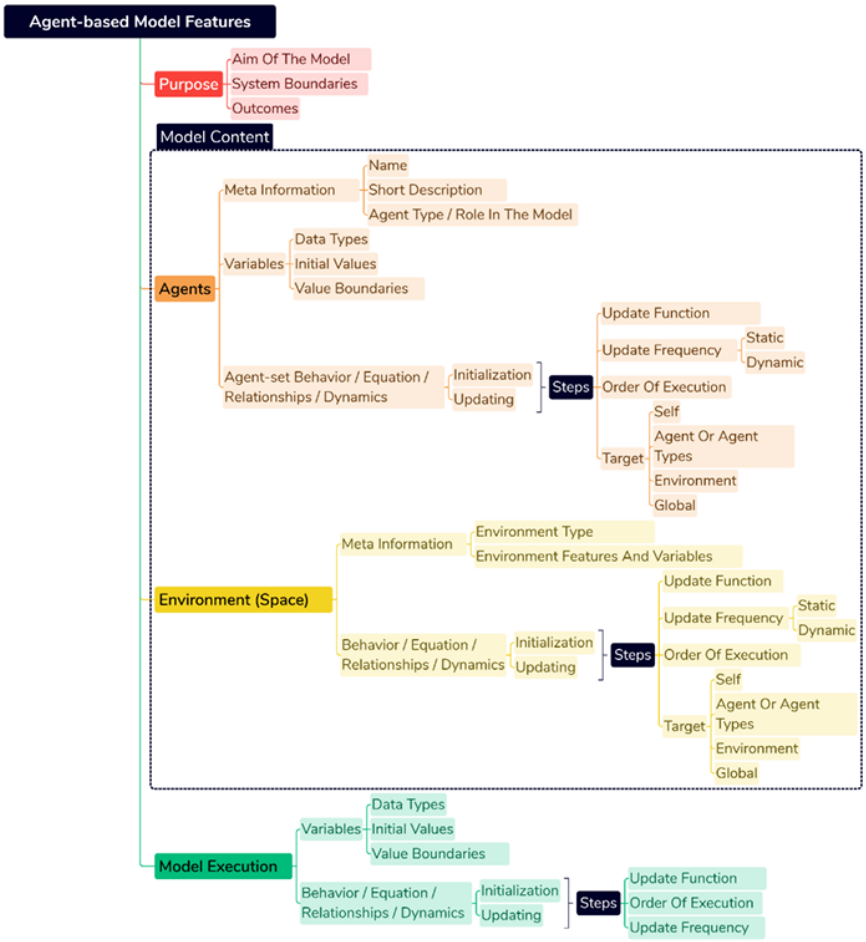

Figure 2: Target information structure, capturing features of the conceptual model as well as implemented model

2.3 Such basic model structure comprises model, environment, and agent-set variables, as well as information regarding the model's purpose. Simulations typically involve variables and mechanisms that are initialized at the beginning and updated at each simulation tick according to the defined mechanism. The initial stratification, shown in Figure 2, categorizes modeling aspects into four general categories: model purpose, agent information, environmental information, and model level information. We believe that the model's purpose is essential for understanding its underlying motivations and premises, and the bottom-up philosophy of agent-based models emphasizes the central role of agent representations in the process. Model information and details about model execution, which provide technical perspective and fine-grained information about the model internals, are often less emphasized. In the following, we provide a more detailed account of these four categories. The model's purpose includes the research questions, system boundaries, and important factors targeted for the study. While this information is not directly involved in the coding process, it is crucial for summarizing and highlighting essential aspects of the model (Müller et al. 2013; Grimm et al. 2020b).

2.4 Agents, their attributes, and their interactions are the most important components of agent-based modeling and their complexity (Edmonds & Meyer 2017; Railsback & Grimm 2019). The interactions and behavioral dynamics create randomness, which is fundamental to ABS. This randomness distinguishes ABS from direct mathematical calculation. When writing code, it is crucial to consider the order of execution, as it directly impacts the outcomes based on the processing of different agents and variables. Besides providing meta information such as name and description and characteristics like data type, initial values, and value boundaries of variables, it is essential to define the behavior function, sequence of updates, and update frequency (e.g., 12 months). In addition, a behavior's target, including self-directed behavior and interactions between different agent sets and the environment, is specified.

2.5 The environmental variables define the simulation space and the environment in which agents are distributed. It is important to note that most secondary assumptions made by programmers occur at this level of informa-

tion (Galán et al. 2009), making it a potential area for narrative inconsistency in the text. There are different types of environments, including grids like single, multi, and hex grids – a rectangle where agents are placed in a 2D or 3D space, continuous space, and network grids. It is crucial to ensure that these assumptions do not deviate the model's purpose and functionality from those initiated by the thematician (theoretician or field expert). Additionally, each agent and the model itself have a set of variables that are initialized and updated at different time steps with varying frequency based on the defined principles of the conceptual model described or implied in the source text.

2.6 Each ABM has a set of variables and behaviors at the model level, which represent the mechanisms of variable changes in specific periods of time or are intended to be accessible by all other programmed components like agents and environment to either control the simulation or facilitate and optimize its coding. The model-level variables are typically higher-level variables that are not directly owned by an agent, but they usually facilitate and parameterize the overall characteristics of the model or hold aggregated information relevant to the entire model. These variables can include information about agent types, the current simulation step, system states, final step, population, report variables, the model's stopping condition, as well as the specification of units of 'ticks', e.g., each time step representing a month and exogenous variables. In these situations, a simulation can assume a model-level auxiliary variable to avoid unnecessary space and memory usage by all agents, which needs to be reflected in the document and indicates the risk of text inconsistency as well.

2.7 Given the list of target information and the potential capabilities of LLMs, in the following section, we will first discuss the principles of QA models in the context of NLP. The general form of QA (Omar et al. 2023) models has been trained around a large corpus of text and, as an input, accepts a set of prompts. However, the technique we use that is relevant to QA models is called knowledge-based Question-answering (KBQA) (Tan et al. 2023), in which, in addition to pre-trained LLMs and prompts, receives additional information in the form of a file or text and tries to apply prompts on that. As a basis, such models rely on a Read-Eval Print Loop (REPL) interaction pattern, in which appropriate output is produced based on the invocation of the language model with the provided prompts considering the context. In this process, QA models allow for iterative refinement of questions and basic contextualization based on earlier responses.

2.8 These categories provide the basis for an interaction protocol with QA models in order to assess their ability to extract this information systematically. This protocol, alongside the benchmark, is described in the following section.

2.9 The introduction of the attention mechanism (Vaswani et al. 2017) has opened up a new evolutionary path towards DL methods, mainly in NLP, where focusing on past evaluated information is a key function, especially if the meanings are context-relevant. The attention mechanism operates under the assumption that each word in the input content carries a different level of importance, which is based on the estimated output. It aims to assign a weight to each word and adjust the outputs based on its positioning within the trained text. However, the latest general generation of the attention mechanism, known as multi-head attention (used in pre-trained LLMs), attempts to simultaneously assign these varying weights to each word. Subsequently, it combines the final results of each parallel process to estimate the output. In this way, LLMs can capture the overall sense of stress and importance of all the words together rather than focusing on individual words during each update. This method not only affects the accuracy of LLMs in problems like classification and sentiment analysis but also more advanced tasks like text summarization and KBQA (Tan et al. 2023; Omar et al. 2023).

2.10 KBQA is a specialized branch of QA models that allows the inclusion of supplementary context information, enabling the model to respond based on both the input query and the provided context. This approach offers significant flexibility compared to traditional NLP methods, which often require fine-tuning or retraining the model with new information. Fine-tuning demands a substantial amount of new data, comparable in size to the original training dataset, to meaningfully influence the model's weights. In contrast, KBQA models can adapt dynamically to new contexts without retraining, making them a more efficient and scalable solution for incorporating domain-specific knowledge. However, models developed as KBQA systems based on the attention mechanism, like ChatGPT, have challenges such as laziness, stability, and model output evaluation difficulties when compared to traditional KBQA (Tan et al. 2023). As a consequence, using techniques to explore the reliability of model performance is central when assessing their performance. One such example is the Monte Carlo simulation, in which an experiment is run multiple times to calculate the statistical characteristics of the experiment, such as precision.

2.11 In this section, we will explore the use of KBQA models to extract information and summarize it in JSON. The KBQA models can be effective in summarizing large amounts of information. Pre-trained QA models (also known as chatbots or conversational AI) have been trained on a large corpus, and their architecture is such that it receives input (known as prompt) and tries to complete the rest (which is the answer). Thus, to use this system, we need to have a clear set of prompts that reflect information discussed in the previous section.

2.12 Based on those prompts, we establish a benchmark and prompt set using the OpenAI ChatGPT-4o model (Section 2.13). This model is easy to use, and their API supports many programming languages. Afterward, we will compare the answers generated by this model with other open source models by performing a sensitivity analysis (controlling the temperature of the model, which can be considered as the innovation and randomness degree of a model) in a repetitive set of iterations (10 runs per each setup) and calculating their cosine similarity to validate the output. Using that mean of all cosine similarity, we will calculate the efficiency of different acceptable similarity thresholds. Due to the absence of established benchmark datasets for ABM question-answering problems (and hence the inability to compare the resulting answers to existing metrics), this paper addresses a two-fold problem, namely a) to establish a baseline protocol for QA model interrogation to extract ABM information using commercial QA, open source QA, and open source pre-trained LLM based QA and, furthermore, b) to establish baseline metrics for a range of selected simulation models.

2.13 The corresponding models will be introduced in Section 2.14 for ABMs (also referred to as simulation models) and Section 2.17 for QA models; all outputs will be compared based on ***cosine similarity*** relative to the desired answers observed based on systematic extraction of relevant features. In addition, a set of randomly generated answers will be audited – due to limited reference points – manually (Section 2.36) to validate the functionality of the cosine similarity in measuring the outputs of the QA models. Finally, the results of the models will be discussed (Section 4), and a conclusion and outlook will be provided.

Benchmark

2.14 In this section, we will discuss the different aspects that are the basis for the established benchmark, including the ABMs,QA models, a baseline QA model (ChatGPT-4o), as well as prompts as input to the model. Afterward, evaluation metrics and results will be discussed in Section 4. The benchmark process is illustrated in Figure 3, and individual steps will be discussed in the following.

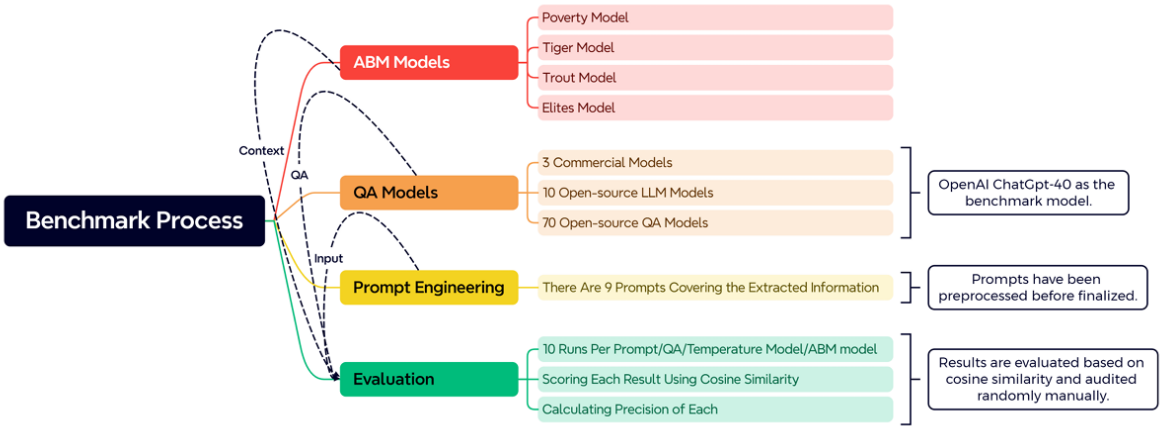

Figure 3: Benchmark Process

ABMs

2.15 A benchmark data set is a collection of input-output data sets that have been gathered and preprocessed for the purpose of training and testing DL models. This allows for a reliable and consistent method of comparison between different model developments and improvements. However, to date, there is no comprehensive benchmark data for ABM information extraction using QA models that cover all protocols and standards as well as all possible up-to-date LLM models. Thus, as shown in Figure 3, in this study, we will use four different ABM models, each with different characteristics and qualities, variably focusing on the level of abstraction, absence or presence of an environment, as well as the nature and depth of its description/specification (selectively relying on ODD or just on textual description). The first model is drawn from the area of economics and explores the macroscopic patterns of economic demand and its postulated effect on poverty (Poverty Model) (Khatami & Frantz 2024a). This model was chosen because of its concrete description, the very abstract conception of

agency, while at the same time only relying on the conceptual description of the model, without drawing on ODD as a specification protocol. The second and third models have been used to showcase the performance based on a structured and systematic account (such as ODD) to specify model details. The models used for this purpose are the tiger (Tiger Model) (Carter et al. 2015) and trout (Trout Model) (Ayllón et al. 2016) models, which are from the ecological field, mainly containing details not only about the agents but auxiliary information such as environmental and model level themselves. Lastly, we drew on a conceptual model of elite knowledge diffusion in a network (Elite Model) (Salimi et al. 2019), which is described in conceptual terms only (i.e., without ODD or a programmatic level of description), and – in contrast to all other models – does not feature an explicit environment.

2.16 These models (illustrated in Figure 4) will serve as an input context (a supplementary document for qa models) for the QA models alongside the input prompt (input task). It is important to note that the results produced by QA models are, by the very nature of the approach, not linguistically deterministic (i.e., provide the identical linguistic output), but rather need to be consistent in concept (i.e., refer to the same concept). In consequence, the responses of the QA models require manual validation at this stage or rely on text similarity techniques, such as cosine similarity (Lahitani et al. 2016; Steck et al. 2024), an approach used in this work.

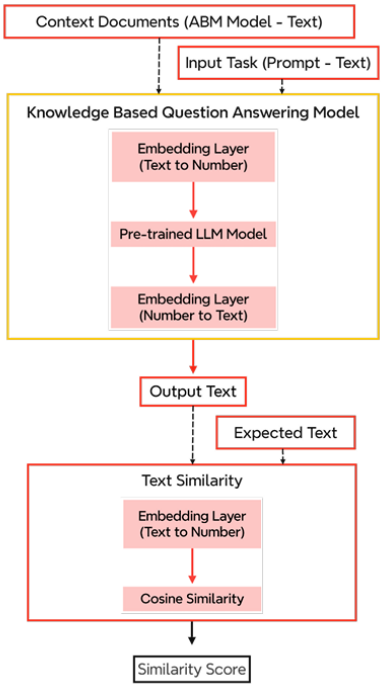

Figure 4: KBQA-Extraction Methodology

2.17 We are drawing on conceptual ABM papers because they are typically written after the model has been created, which reduces the risk of information leakage, and context consistency, although it is not completely eliminated, thanks to the establishment of protocols like ODD to be able to argue the experiments presented in this work. Despite the descriptive nature of ODD, but also to account for variants as well as further development of model specification protocols, the approach proposed here is intentionally applied agnostic of specific protocols to avoid potential overfitting. Overfitting in the field of AI occurs when training and experimental conditions fail to comprehensively represent the full range of real-world possibilities. This issue is commonly associated with limited training datasets or insufficient training iterations. In the context of this study, while the supplementary contexts we provide (such as ODD models) are not part of the core LLM training process, relying solely on ODD models can still lead to overfitting. This is because the narrow focus on ODD models may result in conclusions that do not adequately reflect the complexity and variability of real-world scenarios, thereby limiting the model's generalizability and practical applicability. Moreover, the diverse models enable us to explore to what extent ODD does, in fact, facilitate an automated extraction (i.e., Tiger and Trout models) – in contrast to models described in purely textual form. As a result, we rely on the previously mentioned two models that do not rely on ODD (Poverty and Elites), which address certain aspects covered by ODD and some that are not addressed by ODD.

QA models

2.18 In this study, we will utilize three types of QA models. The first category comprises ready-to-use open source QA models, which can be seamlessly accessed through the HuggingFace API – a platform that distributes a wide range of AI models. The second category includes open source models developed using the LangChain library in Python (Harrison 2023). LangChain enables the integration of pre-trained LLMs as the core of QA tools, allowing us to create systems similar to chatbot frameworks like the ChatGPT family. Given the critical role of the pre-trained LLM in these models, we refer to them as pre-trained open source QA models. The third category consists of commercial models, such as various API versions of ChatGPT, which require financial support for use and offer limited control. By employing these three distinct types of QA models, we aim to explore their effectiveness in supporting the ABM development process.

2.19 As of July 2024, there are approximately 11,000 open source QA models and pre-trained LLM models and more than ten widely used commercial QA models available through corresponding APIs. These models offer different features, including the ability to fine-tune or customize them to specific contexts and content. While using an open source model is generally preferred for control and cost reasons (at least at first glance), fine-tuning and running a powerful LLM can be quite challenging due to the significant budget required, particularly in terms of power and hardware requirements. Typically, the initial cost of fine-tuning and running a complex LLM with different parameters model is about 1,000 times higher than using commercial models served through dedicated servers in small-scale applications. Parameters in machine learning models represent the weights that the model optimizes based on specific input-output relationships. For instance, Llama2, a large language model (LLM), is available in different versions with 7, 13, 34, and 70 billion parameters. These parameters are updated during training through an optimization process, typically involving massive matrix multiplications. This process becomes increasingly complex as parameters are distributed across multiple layers, with each layer's output serving as the input for the next. To update the weights (e.g., using gradient descent), derivatives of the outputs are calculated relative to the inputs. Due to the dependency between layers, derivatives must be propagated back through all preceding layers (using the chain rule), making training both time- and memory-intensive.

2.20 In a simplified scenario, each layer computes weights for a linear regression model $y = wx$, where w is the weight being optimized, y is the output of the layer (and input to the next), and x is the input from the previous layer. However, LLMs, such as Llama2, typically consist of transformer layers, which are more complex. Each transformer layer includes multi-head attention mechanisms that consist of single attention layers (Figure 5). These attention layers contain nonlinear layers: Query (to capture the current focus of the input), Key (to encode the sequence context), and Value (to aggregate information based on attention weights). The outputs of these attention components are concatenated, passed through additional nonlinear and activation layers, and then used to produce the final output of the layer as multi-head attention. This will enable the possibility of a single model to focus on different parts of a text simultaneously.

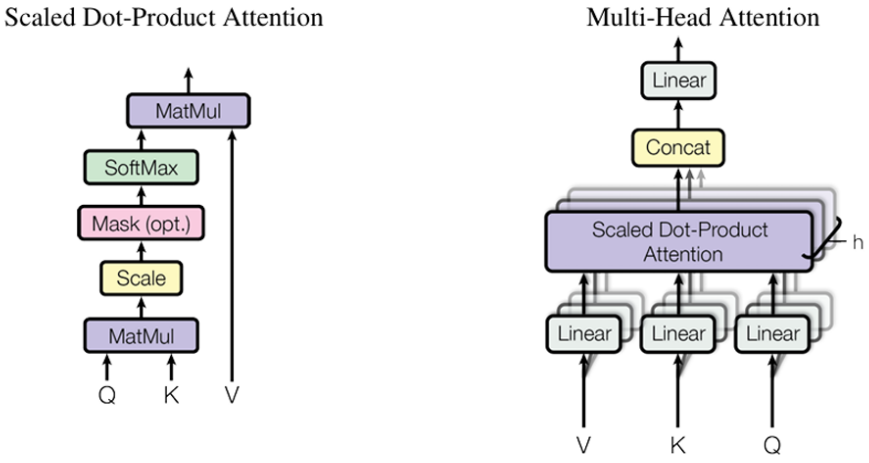

Figure 5: Single vs Multi-head Attention Mechanism from Vaswani et al. (2017)

2.21 For example, Llama2's 7-billion parameter version comprises 32 transformer layers, each with this intricate

architecture. This layered structure and the associated computations significantly contribute to the resource demands of training and deploying LLMs.

2.22 To narrow down the selection of relevant models, we refined the questioning process. Initially, we selected all models with more than 1,000 downloads (as a proxy of utility), rendering us with 70 top-downloaded open source QA models (Downloads were drawn from Hugging Face; HuggingFace Community 2024) and ten open source pre-trained LLM models to make KBQA using LangChain from the Hugging Face QA models repository. Each model was assigned three basic questions, such as the document title, authors, and publication date. This pre-selection of models is necessary due to the high time complexity required to run all mentioned models for all tasks. Before running all prompts (9 prompts) for all selected ABMs, we filtered models by robustness by exposing them to increasingly upcoming complex tasks (the main prompts to extract ABM information). Unfortunately, all open source QA models (Figure 6) failed to answer the simple task, but QAs developed using LLM models (10 models) have had acceptable outputs. However, while meeting the initial baseline goals for QA model inclusion, the models provided challenges with respect to usefulness in the later processing stages, especially with respect to consistency and accuracy when being exposed to more complex queries, as well as output format generation (see Figure 8). Commercial models, on the other hand, showed fewer challenges in this regard, making them more promising. However, from those three commercial models (Figure 6), their latest version (ChatGPT-4o) has been selected to be compared with the open source LLM models.

2.23 Open source LLM models can be considered more accessible than commercial models, and indeed, there are more than ten types of LLMs. However, due to the large size of pre-trained LLM models, and their specific graphic card requirements (like graphic cards and operation systems) it is not possible to run all those on consumer-level GPUs. The light versions of LLM models, for example, llama-2-7b with 7 billion parameters, take up to 1 minute to finish a task using an Nvidia RTX 3080 12G GPU, and for its larger version, the run time takes 4 hours, or, DeepSeek is required to be run on Linux systems with specific graphics cards from the RTX4080 or H100 families. Thus, a set of LLM models (llama-2 7b and 13 b versions, llama-3 8b version, gemma 2b and 7b versions, gemma-2 9b and 27b versions and phi 1, 1.5, 2 versions from Meta, Google, and Microsoft LLM models respectively) have been used (Figure 6). The only difference between LLM-based QA models and commercial QA models with open source QA models is the temperature parameter, which, to some extent, controls the innovation level (randomness) of the model in generating answers. The higher the temperature is, the more variable the answer the LLM will return. In essence, higher temperatures motivate the model to not only narrowly draw on content provided for extraction, but also to draw on its prior knowledge built up as part of the training process, but not immediately relevant to the task at hand. Open source QA models differ from LLM-based and commercial QA models in that they do not utilize a temperature parameter. This distinction arises from the fundamental difference in their methods: open source QA models typically fall under the extractive QA category, whereas LLM-based and commercial models are generative. Extractive models, like the open source ones, aim to predict the most likely span of text that constitutes the answer within the provided context. They do not establish connections between the input query and the broader context beyond this span prediction. If the context and input query are not well-aligned, their outputs may lack coherence. While this could be seen as a limitation, it can also become a strength if the models are fine-tuned on specific datasets, allowing for highly accurate span extraction in specialized contexts.

2.24 Conversely, pre-trained LLM-based QA models, whether open source or commercial, generate answers by leveraging the context, input query, and their training data. These generative models can produce outputs even when relevant information is absent from the provided context, drawing on their pre-existing knowledge. This generative capability is influenced by the temperature parameter, which controls the model's reliance on its training data versus the input context. A higher temperature encourages the model to generate more creative or diverse outputs, relying more heavily on its training data. While prompts can be designed to instruct the model to avoid introducing new concepts, the generated outputs must still be validated to ensure their relevance and accuracy. This balance between extractive precision and generative flexibility highlights the trade-offs between these different types of QA models. Thus, to be able to compare all these models with and without temperature, the variance of cosine similarities will be reported as part of the results for all models (see Figure 9) besides the mean values for all models.

2.25 Given the intent to extract information from given models that reflect the 'context' that the QA model should draw on during its operation, a criterion constraining the choice of candidate models is the ability to provide documents or text to contextualize the operation. To date, there are numerous commercial models available, including Google's Gemini (Team et al. 2023) and OpenAI's ChatGPT (OpenAI et al. 2023), yet only OpenAI models have the ability to capture context in textual form (KBQA). It is essential to clarify that by word "context" we are not referring to the interpretation of the text or any concepts related to its presentation and surrounding information. In LLMs, context refers to a supplementary document that can be included in addition to the main

prompts. The model will then prioritize this additional resource when answering questions. Therefore, three versions of OpenAI models (ChatGPT-3.5, ChatGPT-4.5-turbo, and ChatGPT-4o) have been chosen for the experiments. Among these models, ChatGPT-4o, which is the latest release of OpenAI, was used as a base model for benchmark development and calibration. Figure 6 summarizes the included and excluded models in the current study.

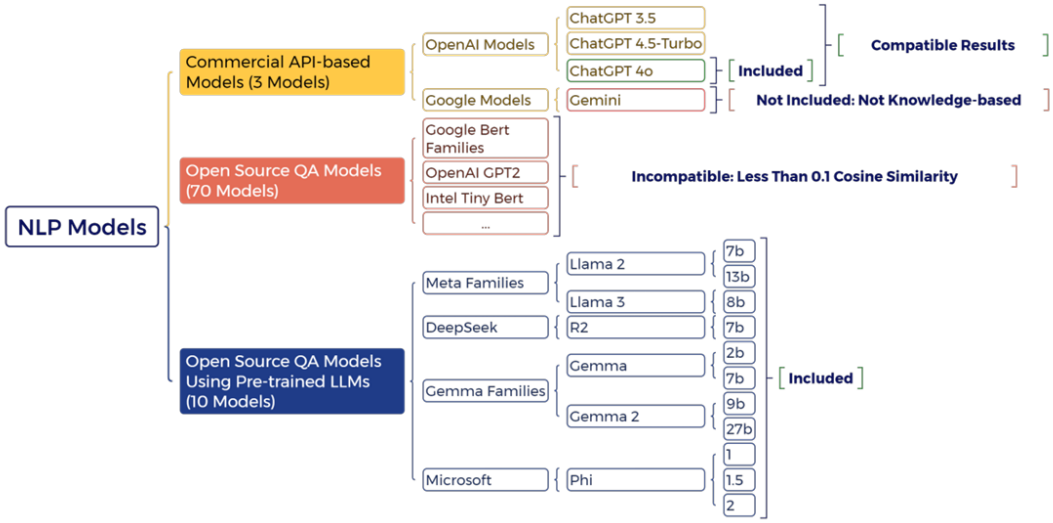

Figure 6: Summary of NLP models included and excluded in the study

Prompt engineering

2.26 Central for the quality of answers provided by QA models is the nature and quality of questions posed to those models. Thus, prompt engineering – giving clear instructions to the model and including all necessary details in the questions – is an important topic of study for researchers (Giray 2023; White et al. 2023), and generally relies on iterative refinement to strike the balance of applicability, i.e., the provision of relevant results for specific cases, and generality, i.e., the usability in a wide range of open/unknown cases.

2.27 One of the difficulties in prompt engineering is checking the effectiveness of the prompt. This issue is not just related to quality of the prompt (e.g., in terms of wording or structuring), but also about handling challenges related to the non-deterministic nature of QA models. Recognizing that both those aspects operate in a trade-off, a central insight is to provide a clear, well-structured, but simple (i.e., not complex) prompt to increase response consistency. For our evaluation, we explored consistency across repeated use of the same prompt 10 times. Establishing the presumed 'correct answer' is the most frequent response, and the consistency is established based on the fraction (10%) of responses carrying this 'correct answer' (which is randomly selected and audited manually with respect to its correctness).

2.28 For instance, if trying to have the model identify variables related to a specific agent-set, it might respond with variables relevant to other categories (e.g., related to the model or environment) with those specific to the agent-set. Executing the query repeatedly, the candidate's correct answer will appear most frequently in the responses (Giray 2023). Relying on the mean of cosine similarities (which will be discussed further later) as the determinant for the correct answer, the variance is used to operationalize the measurement of the consistency of output answers.

2.29 Practical learnings include the fact that the prompt should clearly state the question and not exceed 4096 tokens (words) based on contemporary model limitations. It is furthermore preferable to avoid nested questions. For example, if you need information about agents, their variables, descriptions, initial values, and equations, start by asking for a list of agent sets and their descriptions. Then, iterate over each agent's name and ask for their variables and brief descriptions. Next, within another loop, ask for the initial value, boundaries, data types, and equations for each variable, as well as their update order and frequency. This approach could lead to improved results. The list of prompts can be found in Figure 7, and an example of a full prompt can be found in Appendix A. In addition, a comprehensive list of prompts and their descriptions has been made available online as a supplementary document. This resource is intended to support future studies by providing a detailed reference

for the prompts used in this study, facilitating reproducibility and enabling further exploration of QA models in related research contexts (Khatami & Frantz 2024b).

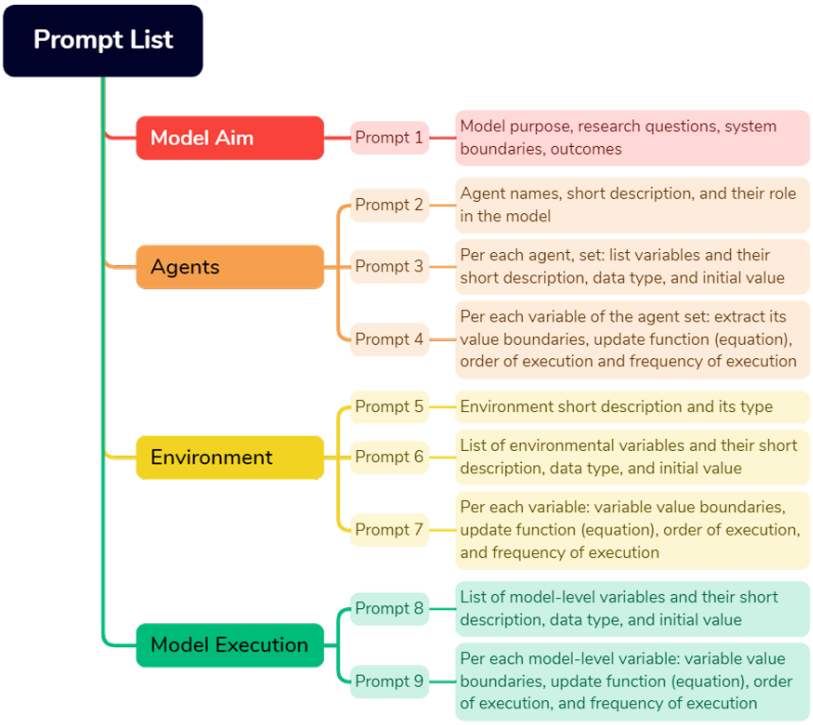

Figure 7: List of Prompts

Results validation

Cosine similarity and embedding layers

2.30 As indicated as part of the Methodology section (Figure 4), a central metric is the ***cosine similarity*** (see Equation 1) which has been used to validate the outputs. Assuming two vectors, A and B, cosine similarity tries to calculate to what extent two vectors (both in terms of attitude and intensity) are similar. The result of cosine similarity is a number between -1 (indicating opposition) and 1 (0-0.25 no similarity, 0.25-0.5 moderate similarity, 0.5-0.75 high similarity, and above 0.75 is an extremely high similarity while below 0 indicates opposite meaning) (Lahitani et al. 2016). However, to be able to apply the cosine similarity, we need to represent our output answers in the form of numerical vectors (Text Similarity in Figure 4). In order to accomplish this, we can use word embedding techniques to convert words into numerical representations that machines can understand (Steck et al. 2024). There are several versions of word embeddings, such as one-hot encoding, TfIdf vectorization, word2vec, word tokenization, and embedding layers based on word tokens, which all transform words into numerical values. Considering the advancements in pre-trained word embeddings and their efficiency compared to other methods (Santosh Kumar et al. 2021; Qi et al. 2018), it has been selected as the numerical vectorization technique.

2.31 Word embeddings are not only integral to the validation process but are also foundational elements of LLMs. In AI, data must be represented numerically to enable optimization processes, and text input is no exception. The embedding process begins with tokenization, where a dictionary is created to map words to numerical tokens derived from the training text. In early DL models, there were attempts to use these tokens directly as input for NLP models. However, unlike image processing—where pixel values are consistently represented as RGB vectors within a fixed range (0-255) and maintain a universal interpretation—text data presents unique challenges.

2.32 When using tokens alone, several issues arise. First, the numerical range can be vast, depending on the number of words in a given language. Second, token counts vary significantly across languages. Third, there is no universal ordering of tokens between different training datasets; for example, the word "the" might be tokenized

as 1 in one dataset but 2 in another despite the fact that token dictionaries are often shared between training and prediction processes to ensure consistency, language-specific discrepancies persist.

2.33 Word embeddings address these challenges by creating dense vector representations of words that capture semantic relationships. Unlike tokenization, embeddings encode words in a multidimensional space where similar words are positioned closer together based on their meaning and usage. This not only standardizes input representation across languages and datasets but also enhances the ability of LLMs to generalize across various linguistic contexts, making embeddings a critical component of modern NLP systems.

2.34 In summary, embedding layers assign a random vector of fixed size N (usually a power of 2 like 32, 64, or 128) to each word in the vocabulary. During the training process, these vectors are refined and updated. As an example, SkipGram is one of the embedding training techniques. The input text is tokenized, and each token (each word) is mapped to its corresponding embedding vector. These vectors are then passed through the layers of the LLM. The idea in SkipGram is to define a window and assume each word gets its meaning from the surrending context in this window. Thus, for example, in the sentences "Decision-maker and autonomous agents are important components of ABM" if the input is "agents" with a window of 6, the output will be ["Decision-maker", "and", "autonomous", "are", important", "components"]. The LLM models with embedding purposes generate predictions, typically aiming to produce text that closely resembles the window considering the input word. The difference between the model's output and the target text is measured, and through a process called backpropagation, the weights in embedding vectors are updated (Wang et al. 2020) up until the training process is finished by achieving an acceptable accuracy of window prediction. SkipGram is just one example of embedding training techniques, and the overarching goal of such methods is to design tasks that map input – output relationships into the model's parameters. Other approaches, such as "next word prediction," involve training the model to analyze a sequence of words and predict the next word in a sentence. These techniques aim to embed meaningful representations of words into vectors that can generalize well across different tasks.

2.35 Building on this logic, modern LLMs often adopt advanced training methodologies like the "Teacher-Student Knowledge Distillation" approach. In this framework, a larger, pre-trained LLM – referred to as the "teacher" – is used to generate labeled input-output pairs for a specific task, such as sentiment analysis. For example, models like BERT (Reimers & Gurevych 2019) can serve as the teacher by providing text inputs along with their corresponding sentiment scores or other labels.

2.36 These labeled datasets are then utilized to train smaller, more lightweight models, referred to as "student" models, such as *all-MiniLM-L6-v2*. The student models learn to approximate the performance of the teacher model while requiring fewer computational resources. By distilling knowledge from the teacher model into the student model, embeddings in the student model are trained more efficiently, enabling them to perform well on downstream tasks with reduced complexity and memory usage. This approach strikes a balance between performance and efficiency, making it particularly valuable for practical applications requiring lightweight models. This iterative process fine-tunes the embeddings, allowing them to capture the semantic and syntactic nuances of words within their contexts. Thus, instead of using each word, an embedding vector is used so a sentence can be presented in the form of a 2d matrix. All-MiniLm-L6-V2 is an LLM model that has been used for the quantification of natural language. It has been trained using a comprehensive natural language data set consisting of different scientific and non-scientific documents to generate a robust embedding layer (Wang et al. 2024), which we will use as part of text similarity analysis.

$$Cosine\ Similarity = \frac{a \cdot b}{||a||\,||b||} \tag{1}$$

Evaluation

2.37 The evaluation of this task involves extracting information from a context (such as a PDF file) and asking questions. To date, this task has to be performed manually since results can firstly contain a mix of qualitative and quantitative information (depending on specific prompts), but are furthermore not linguistically deterministic, i.e., can vary in syntax while capturing the same semantics. Therefore, to ensure the cosine similarity mechanism is functioning as expected, a random small set of answers needs to be manually checked to differentiate between cases in which the intended information present in the underlying simulation is correctly extracted (while recognizing extraction in varying or inconsistent form), or incorrectly extracted (e.g., factually wrong, or not extracted despite evident presence). To control for these artifacts of QA models, we deem a single run insufficient to serve as a reference point. To address this issue, each prompt has been run 10 times, and the mean and variance of cosine similarities have been reported per LLM/ABM/Prompt in Figures 8 and 9.

2.38 Overall, the full picture of the methodology is described in Figure 4), and discussed in this section. Following the introduction of the methodological principles, criteria and considerations for the evaluation of QA model performance, as well as briefly highlighting the selected simulation models, in the upcoming section we will turn to the discussion of results of the systematic interrogation of the selected QA models.

Results

3.1 Following the exploration of various prompt configurations for each target information specified in Section 2, we identified a pattern of prompts that consistently evoked the anticipated responses across the majority of our testing scenarios. To assess the reliability and stability of the model and prompting approach, we executed a minimum of 10 iterations employing the benchmark ABM and QA models. As described in the evaluation section (Section 2.29), each text is converted to a 2d matrix (each vector in this matrix represents a word) using the all-MiniLM-L6-v2 pre-trained LLM model. Afterward, these matrices are used to calculate the cosine similarity between the reference text (expected output per ABM/prompt).

3.2 In order to extract the information described in Section 2, the querying has been divided into themed prompts that reflect specific aspects of the ABM features, e.g., model purpose as Prompt 1, and agent information as Prompts 2-4, etc. The full mapping between ABM features and corresponding prompts is listed in Figure 7 with an example of a full prompt in Appendix A. The provided prompt list is a reflection of the iterative process of refining questions to obtain at least one correct answer using the Poverty model and ChatGPT-4o. As previously discussed, an important insight in achieving better results was to decompose lengthy nested tasks into smaller ones. However, it is important to note that this approach involves trade-offs with respect to economic and sustainability concerns, given the significant energy consumption associated with the use of such models for each query.

3.3 The operational utilization of each QA model was contingent upon a priming task or example that explained the model's role prior to data extraction. Furthermore, operational instructions were provided. The corresponding prompt preceding each query is presented in Listing 6 in Appendix A, and the instructions can be found in Listing 2 in Appendix A. These instructions essentially serve as a guide, outlining the task and expectations to the model.

3.4 Starting from commercial models, ChatGPT-4o outperformed other 3.5 and 4-turbo versions, so only ChatGPT-4o is included in the final results. Moving on to the open source models, there are two versions: pre-trained QA models and LLM-based pre-trained QA models. The major difference between these two groups is that pre-trained QA models are specifically designed for question-answering, while LLM-based pre-trained models have two components. The first component is a separate multi-purpose LLM that can be used for tasks like auto-code generation, and they are generally more robust compared to pre-trained QA models. This LLM is then used in a chained process, in which it memorizes the previous tasks and prompts to make a QA model (known as chatbots). None of the open source QA models (not the LLM-based QAs) identified in the earlier pre-selection were able to handle complex contexts or the specific tasks assigned to them. They either returned irrelevant short answers or indicated the need to fine-tune the models for the specific context. Hence, we removed them from the analysis, rendering this a subject of dedicated research focusing on context-specific pre-training instead of using off-the-shelf models as we did in this work.

3.5 Given all, Figure 8 presents a summary of the precision of different runs for each prompt and LLMs mentioned previously.

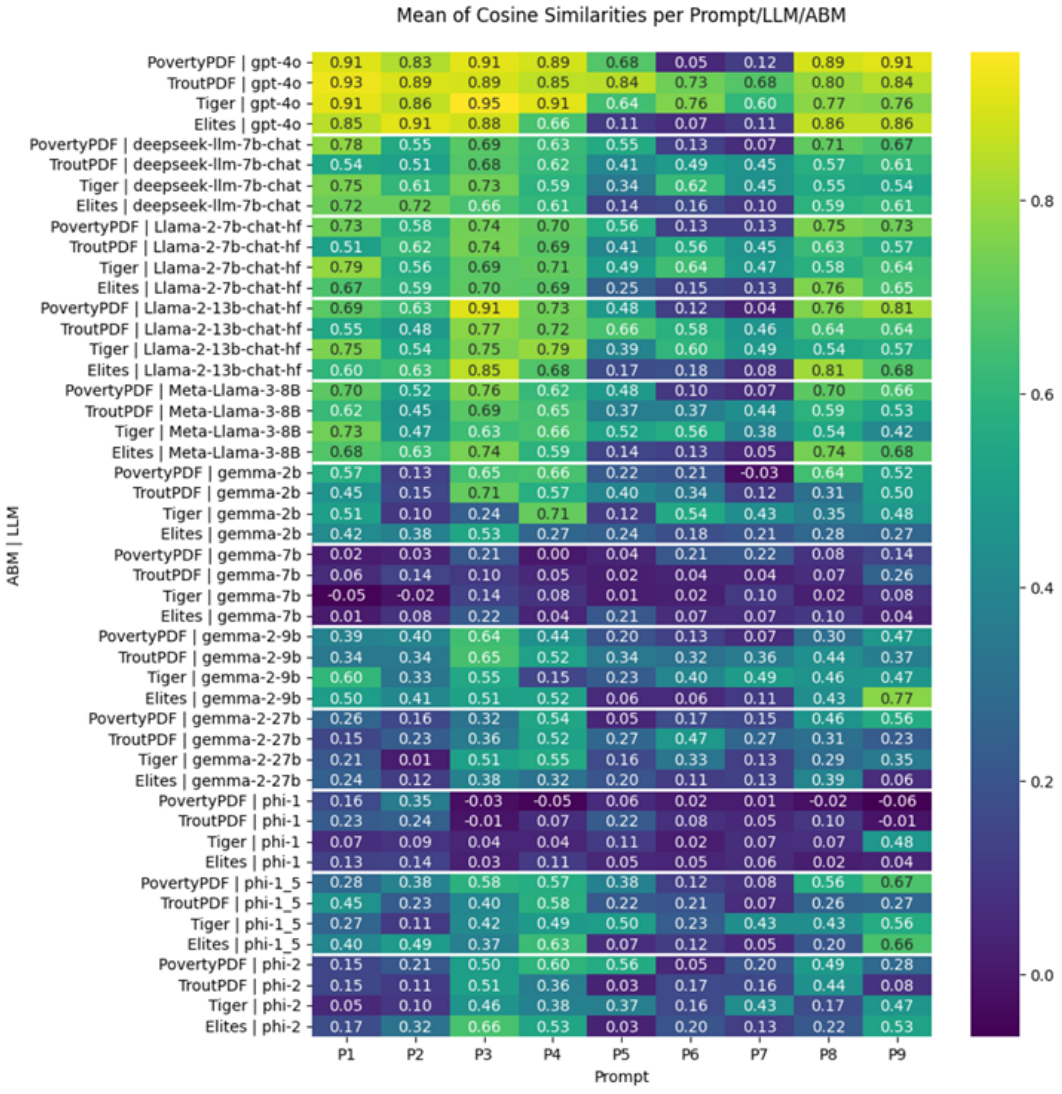

Figure 8: Mean of Cosine Similarities per Prompt/LLM/ABM

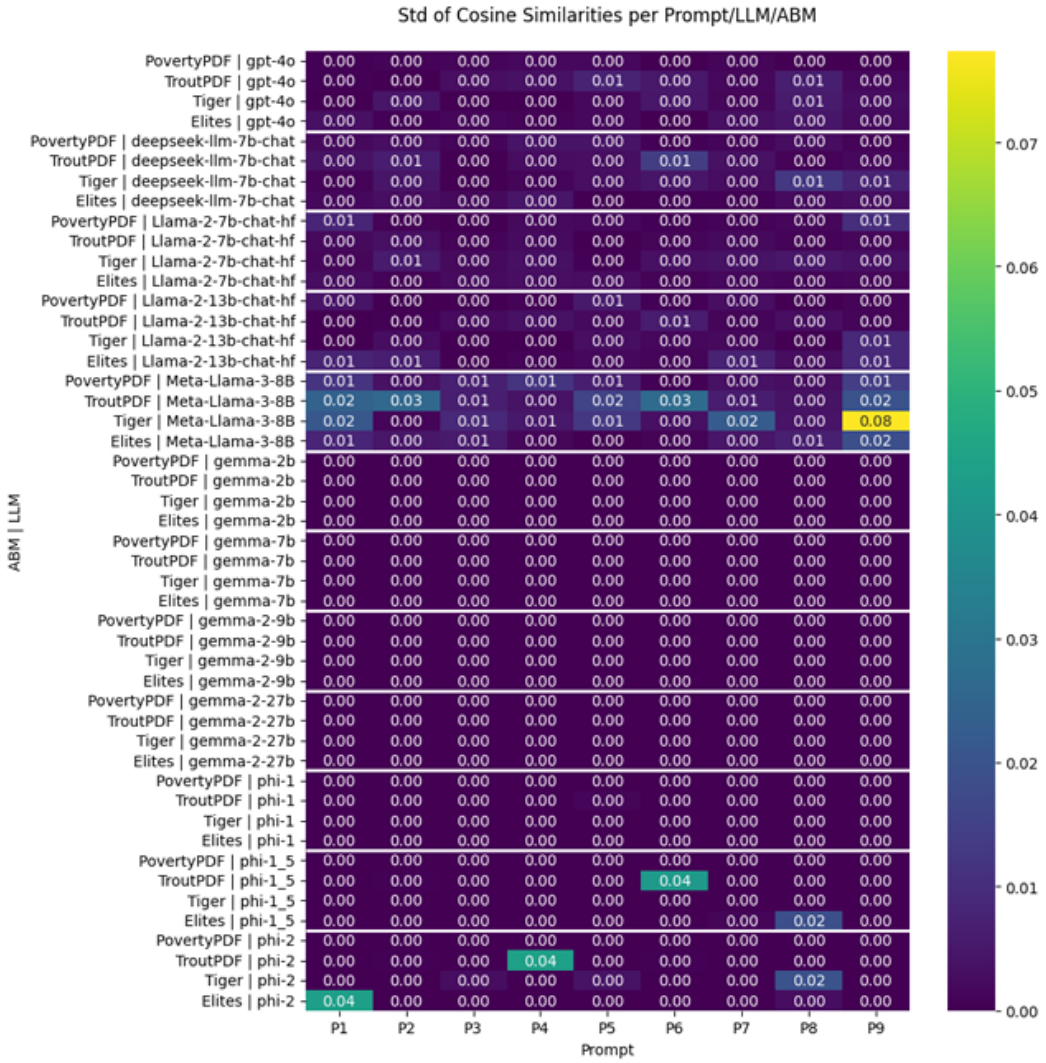

Figure 9: Variance of Cosine Similarities per Prompt/LLM/ABM

3.6 Reviewing the results for the individual QA models across the selected simulation models reveals relevant insights (while the full report of results is uploaded on GitHub repository (Khatami & Frantz 2025),[1] an example of outputs can be found in Appendix B). There is a clear distinction between the results related to environmental prompts (Prompts 5-7) and other prompts. When looking at environment-relevant prompts, all models have either poor or moderate scores (on average, less than 0.5 cosine similarity). Basically, negative values represent opposite meanings. However, similar to the interpretation of positive values, these opposing meanings are more meaningful when they are below -0.25. As the results indicate, although the calculations yield some negative values, they are often close to zero. Consequently, interpreting them as indicative of an opposite meaning is not valid in such cases. Instead, these sentences lack similarity or represent random responses without any meaningful relationship. Upon further examination of ABMs, it becomes evident that not all models include auxiliary assumptions specific to their environment and space. Even in models developed by the ODD protocol, the descriptive focus on other information such as model purpose, model-level variables and equations, agents, and their interactions does not adequately cover the environmental (spatial) relevant information, mainly due to the auxiliary nature of those details. This emphasizes the need for protocols to assist model developers in recognizing and documenting these auxiliary needs as well as the importance of comprehensive protocols like KIDS, as these types of assumptions are not typically brought to light unless a programmer recognizes their importance.

3.7 With regard to other prompts (excluding environmental ones), we observed that open source QA models (red group in Figure 6) were not included in the final results illustration due to their lack of similarity and accu-

racy. However, out of all the commercial models, only ChatGPT-4o has been included. Regarding the results, ChatGPT-4o produces very robust outputs with very high similarity (cosine similarity above 0.8) and low variance (below 0.05). Besides, ChatGPT-4o is capable of extracting information with very high cosine similarity in environmental prompts for ABMs developed considering ODD protocols (cosine similarity above 0.75). This indicates the potential of these model families and expected improvements in future versions to reduce remaining ambiguity.

3.8 In the context of LLM-based open source QA models (the blue category in Figure 6), the Phi family models tested in these experiments either showed no similarity or weak similarity. This was particularly evident for prompts related to agent sets. Although they sometimes indicated high levels of similarity, it was insufficient to draw a general conclusion. On the other hand, the Gemma families, divided into two subgroups - Gemma (gemma-2b and 7b) and Gemma-2 (gemma2-9b and 27b) - showed that gemma 7b, despite having more parameters than gemma-2b, did not yield significantly improved results. This is because a high number of parameters does not necessarily lead to better results (Srivastava 2023). Highlighting the needs to have a balance between model complexity and the given task. However, gemma-2 (9b and 27b versions) has moderate accuracy, which renders all gemma versions with average performance, with similarities averaging around 0.5. This pattern changes when we move from the mentioned models to LLama families (llama3-8b and llama2-7b and 13b have been included only in these experiments mainly due to the high time complexity of other models to be run for these tasks). As the last model, DeepSeek also demonstrates high potential with slightly lower performance ($\tilde{5}$%) compared to Llama 2 7b.

3.9 We can see that all included Llama models, on average, have a very high cosine similarity (above 0.75 on average), making them robust open source models compared to commercial models like ChatGPT-4o. This indicates that these models can be used in projects not only to develop QA models and improve benchmarks but also, due to their pre-trained nature, likely to prove useful as candidates for other tasks like validation and verification of ABM models as well as automated code generation.

3.10 The discussions to this stage were mainly LLM-oriented; however, as mentioned before, a set of different ABM models with different characteristics has been used for the experiments. Considering the results, protocols like ODD not only facilitate the documentation for humans, but its structural layout makes it accessible for LLM models, highlighting the necessity of following those standards.

3.11 As the last point, even though different temperatures and different numbers of runs have been conducted per each pair of experiments, the variance of outputs (Figure 9) illustrates a consistency in the outputs of models.

3.12 The most noteworthy achievement of the ChatGPT-4o and Llama models is their ability to bridge the gap between models supported by ODD and those relying on conceptual descriptions. They accurately detect when an environment specification is missing for the Elite model. These results clearly demonstrate the significant progress that LLM models have made over time. They also show promise in extracting a consistent conceptual structure from ABM descriptions, whether they are based on structured information or on model specifications determined by documentation protocols.

Conclusion

4.1 The main aim of this work is to explore opportunities enabled by novel AI techniques to address the challenges in ABM development that are based on the oftentimes qualitative nature of the modeling process (based on domain and involved stakeholders), with a specific focus on the extraction of model information from conceptual models. At the same time, it is important to bear in mind aspects such as validity and consistency that are central to ensuring robust practices in ABM .

4.2 In this paper, we investigated three categories of Question-Answer models (commercial – 4 models, open source QA models – 70 models, and open source pre-trained LLM-based QA models – 10 models) to help extract relevant model features from their conceptual description. The goal is to summarize the conceptual features in both human and computer-readable format (in a JSON schema) to firstly improve the collaborative process of ABM for both simulationists and non-simulationists, secondly, increase the speed of collaboration in the refinement process of the ABM model development; and thirdly, make it available for auto-code generation efforts (see e.g., Siebers 2025; Bersini 2012; Siebers & Onggo 2014; Apostol et al. 2022).

4.3 We recognize the difficulties faced by non-simulationists in documenting narratives, creating conceptual and simulation models, and evaluating model outputs. Given the rapid developments in this area and the need to analyze those with respect to potential and challenges, instead of offering a singular solution, we propose a

benchmark and roadmap for extracting model information from conceptual models. The reason we picked up the conceptual model (Step 3 in Figure 1) is that the conceptual model provides more consistency than the narratives only, and furthermore, the inclusion and exclusion of narratives in a conceptual model is both a matter of philosophical discussion and complex selection decisions. This, however, demands its own protocol to be developed first. Thus, this approach aims to bridge the gap between conceptual model and simulation model development, which was previously done by utilizing techniques such as ODD standards, UML illustrations, and earlier natural language processing approaches. Our benchmark highlights general ABM features to be extracted, and we apply it to a sample of selected simulation models (reflecting the diversity in feature set and formality), using different QA models to assess their respective performance. The results showcase the general potential of QA models (ChatGPT-4o and Llama families) for the purpose of ABM information extraction while highlighting the challenges encountered by QA models with respect to different ABM features, showcasing the substantive improvements achieved by contemporary models.

Contributions

4.4 To address the overarching objective of this work, we categorized the contributions of this work into three categories. The first contribution is to establish a roadmap for machine-based ABM development (A backbone to facilitate ABM development with Natural Language Processing – Figure 1). This perspective tried to put standards and protocols like ODD and KIDS alongside each other in order to establish a comprehensive map for ABM development (Figure 1). This map highlights the complexity of ABM development due to the involvement of qualitative data and natural language. This complexity is not only revealed in the form of individual errors (ignored or inconsistent documentation, comprehensive evidence collection, etc.) but also seen in communication errors and artifacts (omitted auxiliary assumptions, code optimization, model refinement, etc). Summarization of the information from a conceptual model into an implemented model is the aim after establishing this roadmap due to the refinement loop and highly inconsistent model documentation.

4.5 The second contribution of this work was to design a methodology to facilitate the process defined in Steps 3 and 4 in Figure 1 (i.e., supporting the transition from conceptual model to implemented simulation model). Thus, considering the importance of the human-based validation process, this method should be assessable by both computers and humans. To do so, a natural language approach has been chosen to summarize and convert the conceptual model to a human-readable format (more precisely, a JSON-based schema) which can be processed both by humans and computers for verification and automated code generation. From different natural language available methods, QA models have been selected for this purpose due to their recent improvements, text extraction, and summarization capabilities.

4.6 In order to use QA models, a set of instructions and prompts are required as input for the model. As a consequence, the third contribution of this paper is the definition of a set of 9 prompts that help us summarize the required information for the simulation implementation step. This has been done in an iterative fashion, considering technical developments in different QA model versions. Combined, the proposed prompts, the collected ABM models, as well as the benchmark QA model (ChatGPT-4o), can be used in future research efforts to serve both as a testbed (methodology) as well as baseline (results) for comparative studies that reflect the progression in QA model development.

4.7 A central challenge of this approach is the nature of the data: inputs as well as generated and expected outputs of QA models are provided in the form of qualitative data (natural language), which makes the validation process challenging. Addressing this, we considered a broad range of metrics in deep learning and natural language processing (accuracy for classification, mean squared error for regression, cosine similarity for text similarity, etc.). Assessing similarity, we draw on cosine similarity as one of the techniques that is commonly applied in the context of text-based comparisons, which, in contrast to a manual validation process, largely automates aspects such as validation and, hence, the overall process.

4.8 In summary, our results show that, first, the current state-of-the-art improvement in LLM models, such as Llama2 and 3 versions (open source) and ChatGPT-4o(commercial), show great promise in automating information extraction and summarization. The outputs show that even though LLMs are showing high potential (specifically with respect to de facto standards such as ODD for the documentation and the KIDS protocol for including all assumptions and narratives), such as auxiliary assumptions, like simulation relevant environmental descriptions, play a significant role, in the quality of outputs. In addition, by manually auditing 10% of generated outputs of models and the expected ones, both robustness and consistency of the automated cosine similarity method have been assessed, reinforcing our beliefs that the methodology mentioned is a sound basis for evaluation and for serving as a comparative baseline.

Limitations and outlook

4.9 Inasmuch as we have developed a foundational benchmark methodology to support the development of agent-based simulation using QA models, there are several limitations that need to be addressed.

4.10 The primary limitation is the absence of a comprehensive benchmark ABM dataset containing both inputs and expected outputs. Creating such a dataset could offer new insights and valuable information about the impacts or norms of model specifications within or across disciplines. It could also assist in establishing improved documentation standards to make models more accessible for analysis. However, this would necessarily afford careful creation and quality assurance associated with such a dataset. An important implication (as it is for this work) would be the careful analysis of such datasets by a variety of researchers reflecting the role diversity of individuals involved in ABM, an aspect discussed at the beginning of this article.

4.11 Furthermore, the rapid progression of QA models poses another challenge. Developing lighter and more robust models, especially open source ones, is essential. One significant challenge we encountered was the inability to fine-tune open source models without substantial pre-training. Conversely, the practical constraint of contemporarily limited availability of graphical computational units hinders the use of more robust versions of pre-trained models like Llama3-70b. Consequently, fine-tuning the recommended list of open source models is an important area for future research.

4.12 The defined roadmap includes steps that involve qualitative data and natural language, which can be processed using NLP techniques. However, due to the fuzzy boundaries of these steps, it is important to further discuss and agree upon the conceptual philosophy, technical methodology, and requirements and protocols for each of those steps. When considering the steps outlined in Figure 1, it is evident that there is a significant gap with different roots (such as comprehensive analysis of narratives, model verification, and validation, as well as model re-usability) that need to be considered in future works.

4.13 Similarly, the proposed features of agent-based models primarily focus on structural aspects. While the extraction of algorithmic features is addressed by the current set of prompts, future work is needed to refine those further to assess their generality and limitations, given that the primary focus in agent-based models builds on the interactionist metaphor that tightly links agents with the social and (simulated) physical environment. This can, on the one hand, be performed by assessing a richer set of exemplary models that feature greater behavioral diversity, but also by introducing nested structures that help extract algorithmic features on a more fine-granular level that enables the reconstruction of both functions and execution cycles for richer sets of heterogeneous entities and the model at large.

4.14 Turning to a more general perspective, this paper serves as a 'prompt' to invite for a more in-depth discussion about the potentials, opportunities, and risks of using QA models in ABM, and whether it is suitable for extraction or generation purposes, the associated challenges, as well as implication of models developed with partial (or eventually full) support of generative machinery. At this stage, the more immediate goal of this work is to support discussions on this topic by proposing general techniques to evaluate the advantages of new analytical approaches while acknowledging the challenges of operating non-deterministic black-box systems. Notwithstanding the need for further engagement, we can, however, posit that the development of ABMs is indeed a practical use case for the use of generative AI.

Appendix A: Prompts Example

```
Please analyze the provided ABM text to identify the purpose of the model. Present the extracted
   data exclusively in JSON format, ensuring that the JSON object is comprehensive and contains all
   requested information. Avoid any form of data truncation or summarizing, and ensure that the
   response is strictly limited to the JSON object without any supplementary text. The JSON should
   follow this structure: {'Model_Purpose': {'full_description':Full_DESCRIPTION, '
   research_questions': ['RESEARCH_QUESTION_1', 'RESEARCH_QUESTION_2', ... ], 'system_boundaries':
   [], 'outcome_variables': {VAR1_NAME: SHORT_DESCRIPTION, VAR2_NAME: SHORT_DESCRIPTION, ....}}}
```

Listing 1: Prompt Example

```
You are an Agent based modeling specialist. Your duty is to help users in extracting information
   from ABM texts for coding purposes. Get the user messages to extract the relevant information
   from uploaded file. Do not summarize information and return a full report of the expected
   information. Return expected information just in the JSON format without any extra text around it
   .
```

Listing 2: QA Instruction

Appendix B: Generated Output Examples

In this section, we provide an example of the expected output for Poverty ABM using Prompt 1. Additionally, we present three different outputs generated with temperatures of 0.1, 0.5, and 0.9 using GPT4-0. The cosine similarity between the expected output and the three generated outputs is 0.891, 0.927, and 0.936, respectively.

Expected Output for Prompt 1, Poverty ABM

```
{
  "Model_Purpose": {
    "full_description": "The primary objective of this study is to use Agent-Based Modeling (ABM) to examine the correlation between income levels and poverty levels by simulating the complex demand behavior of households in the market. The model aims to provide a detailed analysis of economic factors that affect poverty and to facilitate deeper investigations into how income levels and price levels correlate, potentially leading to increased inflation and poverty. Additionally, the model seeks to offer fundamental analytical value by aligning its output with empirical data and discussing future extensions to cover a broader spectrum of economic complexities.",
    "research_questions": ["How do income levels influence poverty levels in a nation?", "What is the correlation between income and the price of a basket of goods in determining Poverty Lines (PL)?", "How does household demand behavior in the market affect macroeconomic outcomes related to poverty?"],
    "system_boundaries": [
    "The model focuses on the demand-side behavior of individuals with assumptions of limitless supply and a fixed price point.",
    "The supply subsystem and government are assumed to be exogenous systems."],
    "outcome_variables": {
      "income_mean": "The average income level of agents (households) in the simulation, directly influencing their demand.",
      "citizen_desired_demand": "The preferred level of consumption by agents, representing the value of the basket of goods.",
      "current_demand_level": "The actual level of consumption achieved by agents, indicating the satisfaction of their demand.",
      "savings": "The amount of income saved by agents after meeting their demand, divided into regular savings and essential investment savings (EIG).",
      "disposable_income": "The remaining income of agents after subtracting their current demand level, used to measure the potential for increased consumption or savings."
    }
  }
}
```

Listing 3: Prompt Example

GPT4-0, Poverty model, Temperature = 0.1

```
{
  "Model_Purpose": {
    "full_description": "The primary objective of this study is to examine the correlation between income levels and poverty levels in a nation using Agent-based modeling (ABM). The model aims to complement established statistical models of poverty by exploring causal linkages that statistical models are poorly equipped to offer. It focuses on understanding the demand-side behavior of individuals in a society, with initial assumptions of limitless supply and a fixed price point, to approximate conditions found in open economies during periods of economic stability. The model seeks to capture distinctive empirically observed growth patterns and provide insights into the potential underlying processes of economic development and associated poverty levels.",
    "research_questions": [
    "What is the correlation between income levels and the price of a basket of goods, serving as an indicator of the Poverty Lines (PL) in a given country?",
    "How do market dynamics, reflecting both demand- and supply-side behavior, influence market prices and poverty levels?","What are the causal determinants that drive a country\'s progression out of poverty?"
    ],
    "system_boundaries": [
    "The model initially focuses on the demand-side behavior of individuals with assumptions of limitless supply and a fixed price point.",
    "The supply subsystem and government rely on external processes that are not included in this model."
    ],
    "outcome_variables": {
      "income_mean": "Represents the average income level of agents, parameterized by different income functions.",
      "demand_level": "Represents the level of demand based on income and desired demand, used to calculate the demand gap.",
      "savings": "Represents the amount of income saved after meeting the demand level, including regular savings and essential investment savings."
    }
  }
}
```

Listing 4: Prompt Example

GPT4-0, Poverty model, Temperature = 0.5

```
{
  "Model_Purpose": {
    "full_description": "The agent-based model (ABM) aims to examine the complex demand behavior of households in the market to establish an experimental platform for investigating the foundations that drive or mitigate economic growth and poverty. The model explores the correlation between income levels and poverty levels, aiming to provide insights into the causal linkages that statistical models are less equipped to offer. It focuses on the demand-side behavior of individuals in a society, assuming limitless supply and fixed price points, to approximate conditions in open economies during periods of economic stability. The model uses stylized income functions to simulate different economic growth scenarios and to capture distinctive empirically observed growth patterns.",
    "research_questions": [
    "What is the correlation between income levels and poverty levels in a nation?",
    "How do demand-side behaviors in a society affect market prices and poverty levels?","Can agent-based modeling provide insights into the causal determinants of a country\'s progression out of poverty?"
    ],
    "system_boundaries": [
    "The model focuses on the demand-side of the market, assuming limitless supply and fixed price points.",
    "It does not include the supply subsystem or government processes, which are abstracted.",
    "The model is based on a simplified market model with a demand-side focus, without direct interaction between agents."
    ],
    "outcome_variables": {
      "income_mean": "Represents the average income level derived from income functions.",
      "income_std": "Standard deviation of income, used to distribute income levels among agents.",
      "demand_gap": "The difference between desired demand and current demand level, influencing demand inflow.",
      "savings": "Represents the excess income saved after meeting demand levels.",
      "investment_savings_state": "Indicates the state of savings allocated for essential investment goods."
    }
  }
}
```

Listing 5: Prompt Example

GPT4-0, Poverty model, Temperature = 0.9

```
{
  "Model_Purpose": {
    "full_description": "The primary objective of this study is to examine the correlation between income levels and poverty levels in a nation using Agent-based modeling (ABM), focusing on the demand side of the market. This approach allows for exploring the complex dynamics of household demand behavior to understand the foundations that drive or mitigate economic growth and poverty. The model provides insights into the fundamental processes that bring about macro-level outcomes, helping to identify causative linkages that are difficult to discern through traditional statistical models alone.",
    "research_questions": [
    "What is the correlation between income levels and poverty levels in different nations?",
    "How do demand behavior and economic conditions interact to affect poverty lines?","What are the causal determinants that drive a country\'s progression out of poverty?"
    ],
    "system_boundaries": ["The focus is primarily on the demand-side behavior within a nation\'s economy, assuming an open economy with periods of economic stability and constant price levels.",
    "The model does not include external processes such as supply subsystem and government interventions, which are abstracted from the model."
    ],
    "outcome_variables": {
      "income_mean": "A variable determining the average income level of agents, used to assess economic scenarios.",
      "demand_level": "Reflects the demand gap and the tendency towards increasing or lowering demand based on the income level.",
      "savings": "The model includes regular and essential investment savings, which influence demand satisfaction and future demand adjustments."
    }
  }
}
```

Listing 6: Prompt Example

Notes

[1]The code base, inputs, and outputs can be found under the following URL: `https://github.com/siamak-khatami/QA-ABM-Extraction`

References

An, L., Grimm, V. & Turner II, B. L. (2020). Editorial: Meeting grand challenges in agent-based models. *Journal of Artificial Societies and Social Simulation*, *23*(1), 13

Apostol, D. C., Rusovan, P. D. & Marcu, M. (2022). UML to code, and code to UML, a view inside implementation challenges and cost. 2022 26th International Conference on System Theory, Control and Computing, ICSTCC 2022 - Proceedings

Ayllón, D., Railsback, S. F., Vincenzi, S., Groeneveld, J., Almodóvar, A. & Grimm, V. (2016). InSTREAM-Gen: Modelling eco-evolutionary dynamics of trout populations under anthropogenic environmental change. *Ecological Modelling*, *326*, 36–53

Bersini, H. (2012). UML for ABM. *Journal of Artificial Societies and Social Simulation*, *15*(1), 9

Carter, N., Levin, S., Barlow, A. & Grimm, V. (2015). Modeling tiger population and territory dynamics using an agent-based approach. *Ecological Modelling*, *312*, 347–362

Chen, J., Lu, X., Rejtig, M., Du, D., Bagley, R., Horn, M. S. & Wilensky, U. J. (2024). Learning agent-based modeling with LLM companions: Experiences of novices and experts using ChatGPT & NetLogo chat. Conference on Human Factors in Computing Systems - Proceedings

Chen, J. & Wilensky, U. (2023). Chatlogo: A large language model-driven hybrid natural-programming language interface for agent-based modeling and programming. Available at: `https://arxiv.org/abs/2308.08102v1`

Edmonds, B. (2012). Context in social simulation: Why it can't be wished away. *Computational and Mathematical Organization Theory*, *18*(1), 5–21

Edmonds, B. (2015a). A context-and scope-sensitive analysis of narrative data to aid the specification of agent behaviour. *Journal of Artificial Societies and Social Simulation*, *18*(1), 17

Edmonds, B. (2015b). Using qualitative evidence to inform the specification of agent-based models. *Journal of Artificial Societies and Social Simulation*, *18*(1), 18

Edmonds, B. & Meyer, R. (2017). *Simulating Social Complexity - A Handbook*. Cham: Springer International Publishing

Edmonds, B. & Moss, S. (2005). From KISS to KIDS - An 'anti-simplistic' modelling approach. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)

Galán, J. M., Izquierdo, L. R., Izquierdo, S. S., Santos, J. I., del Olmo, R. & López-Paredes, A. (2013). Checking simulations: Detecting and avoiding errors and artefacts. In B. Edmonds & R. Meyer (Eds.), *Simulating Social Complexity*, (pp. 95–116). Cham: Springer International Publishing

Galán, J. M., Izquierdo, L. R., Izquierdo, S. S., Santos, J. I., del Olmo, R., López-Paredes, A. & Edmonds, B. (2009). Errors and artefacts in agent-based modelling. *Journal of Artificial Societies and Social Simulation*, *12*(1), 1

Galan, J. M. & Pavón, J. (2009). Reducing the modeling gap: On the use of metamodels in agent-based simulation. 6th Conference of the European Social Simulation Association (ESSA 2009)

Giabbanelli, P. J. (2023). GPT-based models meet simulation: How to efficiently use large-scale pre-trained language models across simulation tasks. Proceedings of the Winter Simulation Conference

Gilbert, N. & Troitzsch, K. G. (2005). *Simulation for the Social Scientist*. Maidenhead: Open University Press

Giray, L. (2023). Prompt engineering with ChatGPT: A guide for academic writers. *Annals of Biomedical Engineering*, *51*(12), 2629–2633

Grimm, V., Berger, U., Bastiansen, F., Eliassen, S., Ginot, V., Giske, J., Goss-Custard, J., Grand, T., Heinz, S. K., Huse, G., Huth, A., Jepsen, J. U., JNørgensen, C., Mooij, W. M., Müller, B., Pe'er, G., Piou, C., Railsback, S. F., Robbins, A. M., Robbins, M. M., Rossmanith, E., Rüger, N., Strand, E., Souissi, S., Stillman, R. A., VabNø, R., Visser, U. & DeAngelis, D. L. (2006). A standard protocol for describing individual-based and agent-based models. *Ecological Modelling*, *198*(1), 115–126

Grimm, V., Berger, U., DeAngelis, D. L., Polhill, J. G., Giske, J. & Railsback, S. F. (2010). The ODD protocol: A review and first update. *Ecological Modelling*, *221*(23), 2760–2768

Grimm, V., Johnston, A. S., Thulke, H. H., Forbes, V. E. & Thorbek, P. (2020a). Three questions to ask before using model outputs for decision support. *Nature Communications*, *11*(1), 1–3

Grimm, V., Railsback, S. F., Vincenot, C. E., Berger, U., Gallagher, C., DeAngelis, D. L., Edmonds, B., Ge, J., Giske, J., Groeneveld, J., Johnston, A. S. A., Milles, A., Nabe-Nielsen, J., Polhill, J. G., Radchuk, V., Rohwäder, M. S., Stillman, R. A., Thiele, J. C. & Ayllón, D. (2020b). The ODD protocol for describing agent-based and other simulation models: A second update to improve clarity, replication, and structural realism. *Journal of Artificial Societies and Social Simulation*, *23*(2), 7

Gurcan, O. (2024). LLM-augmented agent-based modelling for social simulations: Challenges and opportunities. *Frontiers in Artificial Intelligence and Applications*, *386*, 134–144

Harrison, S. (2023). LangChain: Building applications with LLMs through composability. Available at: `https://pypi.org/project/langchain/`

Howick, S., Megiddo, I., Khanh, L., Nguyen, N., Wurth, B., Kazakov, R., Howick, S., Megiddo, I., Nguyen, L. K. N., Wurth, B., Kazakov, R., Nguyen, L. K. N., Wurth, B. & Kazakov, R. (2024). Combining SD and ABM: Frameworks, benefits, challenges, and future research directions. In M. Fakhimi & N. Mustafee (Eds.), *Hybrid Modeling and Simulation*, (pp. 213–244). Berlin Heidelberg: Springer

HuggingFace Community (2024). Models - Hugging Face. `https://huggingface.co/models`

Khatami, S. & Frantz, C. (2023). Copatrec: A correlation pattern recognizer Python package for nonlinear relations. *SoftwareX*, *23*

Khatami, S. & Frantz, C. (2024a). Income versus demand: Exploring dynamics of poverty lines using agent-based modeling. In C. Elsenbroich & H. Verhagen (Eds.), *Advances in Social Simulation*, (pp. 353–372). Cham: Springer International Publishing

Khatami, S. & Frantz, C. (2024b). Prompt engineering guidance for conceptual agent-based model extraction using large language models. Available at: `https://arxiv.org/abs/2412.04056v1`

Khatami, S. & Frantz, C. (2024c). Toward automating agent-based model generation: A benchmark for model extraction using question-answering techniques. Social Simulation Conference

Khatami, S. & Frantz, C. (2025). QA-ABM-extraction: A code base for quality assurance in agent-based modeling. Available at: `https://github.com/siamak-khatami/QA-ABM-Extraction`

Khurana, D., Koli, A., Khatter, K. & Singh, S. (2023). Natural language processing: State of the art, current trends and challenges. *Multimedia Tools and Applications*, *82*(3), 3713–3744

Lahitani, A. R., Permanasari, A. E. & Setiawan, N. A. (2016). Cosine similarity to determine similarity measure: Study case in online essay assessment. Proceedings of 2016 4th International Conference on Cyber and IT Service Management, CITSM 2016

Lan, Y., He, G., Jiang, J., Jiang, J., Zhao, W. X. & Wen, J. R. (2021). A survey on complex knowledge base question answering: Methods, challenges and solutions. IJCAI International Joint Conference on Artificial Intelligence

Liu, C., Bao, X., Zhang, H., Zhang, N., Hu, H., Zhang, X. & Yan, M. (2024). Guiding ChatGPT for better code generation: An empirical study. 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2024

Liu, J., Xia, C. S., Wang, Y. & Zhang, L. (2023a). Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation. Advances in Neural Information Processing Systems

Liu, Z., Tang, Y., Luo, X., Zhou, Y. & Zhang, L. F. (2023b). No need to lift a finger anymore? Assessing the quality of code generation by ChatGPT. *IEEE Transactions on Software Engineering*, *50*(6), 1548–1584

Müller, B., Bohn, F., Dreßler, G., Groeneveld, J., Klassert, C., Martin, R., Schlüter, M., Schulze, J., Weise, H. & Schwarz, N. (2013). Describing human decisions in agent-based models - ODD+D, an extension of the ODD protocol. *Environmental Modelling and Software*, *48*, 37–48

Omar, R., Mangukiya, O., Kalnis, P. & Mansour, E. (2023). ChatGPT versus traditional question answering for knowledge graphs: Current status and future directions towards knowledge graph chatbots. Available at: https://arxiv.org/abs/2302.06466

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J. & et al (2023). GPT-4 Technical Report

Padilla, J. J., Shuttleworth, D. & O'Brien, K. (2019). Agent-based model characterization using natural language processing. Proceedings of the Winter Simulation Conference

Paudel, R. & Ligmann-Zielinska, A. (2023). A largely unsupervised domain-independent qualitative data extraction approach for empirical agent-based model development. *Algorithms*, *16*(7), 338

Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H. & Riedel, S. (2019). Language models as knowledge bases? EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference

Qi, Y., Sachan, D. S., Felix, M., Padmanabhan, S. J. & Neubig, G. (2018). When and why are pre-trained word embeddings useful for neural machine translation? NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference

Railsback, S. F. & Grimm, V. (2019). *Agent-Based and Individual-Based Modeling: A Practical Introduction*. Princeton, NJ: Princeton University Press

Reimers, N. & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-Networks. EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference

Salimi, K., Frydenlund, E., Padilla, J. J., Haaland, H. & Wallevik, H. (2019). The role of elites in the diffusion of social norms of humanitarianism. *Simulation Series*, *51*(1)

Santosh Kumar, P., Yadav, R. B. & Dhavale, S. V. (2021). A comparison of pre-trained word embeddings for sentiment analysis using deep learning. *Advances in Intelligent Systems and Computing*, *1165*, 525–537

Shuttleworth, D. & Padilla, J. (2022). From narratives to conceptual models via natural language processing. Proceedings of the Winter Simulation Conference

Siebers, P.-O. (2025). Exploring the potential of conversational AI support for agent-based social simulation model design. *Journal of Artificial Societies and Social Simulation*, *28*(3), 2

Siebers, P.-O. & Onggo, B. (2014). Graphical representation of agent-based models in operational research and management science using UML. Operational Research Society Simulation Workshop

Srivastava, V. (2023). Classification of fish species using deep learning models. Available at: https://hdl.handle.net/11250/3079160

Steck, H., Ekanadham, C. & Kallus, N. (2024). Is cosine-similarity of embeddings really about similarity? WWW 2024 Companion - Companion Proceedings of the ACM Web Conference

Sterman, J. D. (2000). *Business Dynamics: Systems Thinking and Modeling for a Complex World*. Boston, MA: McGraw-Hill

Tan, Y., Min, D., Li, Y., Li, W., Hu, N., Chen, Y. & Qi, G. (2023). Can ChatGPT replace traditional KBQA models? An in-depth analysis of the question answering performance of the GPT LLM family. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)

Team, G., Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., Silver, D., Johnson, M., Antonoglou, I., Schrittwieser, J., Glaese, A., Chen, J., Pitler, E., Lillicrap, T., Lazaridou, A., Firat, O. & et al (2023). Gemini: A Family of Highly Capable Multimodal Models

Vaithilingam, P., Zhang, T. & Glassman, E. L. (2022). Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. Conference on Human Factors in Computing Systems - Proceedings

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N. & Zhou, M. (2020). MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. *Advances in Neural Information Processing Systems*, *2020-Decem*

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N. & Zhou, M. (2024). HuggingFace Sentence Transformers All-MiniLM-L6-V2. Available at: `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J. & Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with ChatGPT. *Annals of Biomedical Engineering*, *51*(12)