CN7050 – Intelligent Systems Coursework

AI Script Writer: A Multi-Agent Intelligent Content Generation System

Student Name: _____

Student ID: _____

Submission Date: _____

Application and Workflow Selection

1.1 Introduction & Motivation

Scriptwriting is a resource-intensive and cognitively complex activity that underpins content creation across domains such as digital marketing, entertainment, education, and corporate communications. Organizations increasingly depend on scripted material for advertisements, explainer videos, training programmers, and social media campaigns to maintain consistent messaging and audience engagement. However, manual script development is frequently constrained by limited time, high production costs, and creative fatigue, particularly when content must be produced at scale.

Recent advancements in large language models have enabled automated text generation at an unprecedented level of speed and volume. While these technologies demonstrate strong linguistic capabilities, many existing AI-based writing solutions are implemented as monolithic systems that offer limited transparency, configurability, or user control over the generation process. As a result, the internal reasoning behind generated content is often opaque, which can reduce user trust—especially in professional and academic environments where content quality, accountability, and ethical use are critical considerations.

This project addresses these challenges by proposing an AI Script Writer system grounded in agentic artificial intelligence principles. By decomposing the scriptwriting workflow into a set of specialized and cooperating agents, the system supports structured content generation, enhanced quality control, and greater user oversight. This agent-based approach not only improves interpretability and modularity but also enables more effective collaboration between human users and AI systems in creative decision-making processes.

1.2 Explanation of the Workflow + Diagram

The AI Script Writer adopts a modular, agent-oriented workflow coordinated by a central **Coordinator Agent**, which oversees the execution of all analytical components. Each agent is assigned responsibility for a specific stage of the scriptwriting process and returns its intermediate outputs to the coordinator, which controls execution sequencing and aggregates results into a coherent final output.

System Architecture:

The system architecture is centered around an orchestration component that interfaces with multiple specialized agents. These agents are responsible for distinct creative and analytical functions, including user intent interpretation, outline generation, content drafting, stylistic adjustment, and iterative refinement. This architectural design enables clear separation of concerns and supports modular system evolution.

Workflow Diagram:

The workflow is initiated by user input specifying the script topic, target audience, and desired tone or style. Subsequent agents progressively transform this high-level input into a structured narrative outline, a complete script draft, and a refined final version that aligns with the specified constraints and stylistic requirements.

Detailed Agent Flow:

At each stage of execution, agent outputs are explicitly logged and reviewed before the next agent is invoked. This sequential execution model enhances transparency, traceability, and auditability, allowing both developers and users to understand how intermediate decisions contribute to the final generated script.

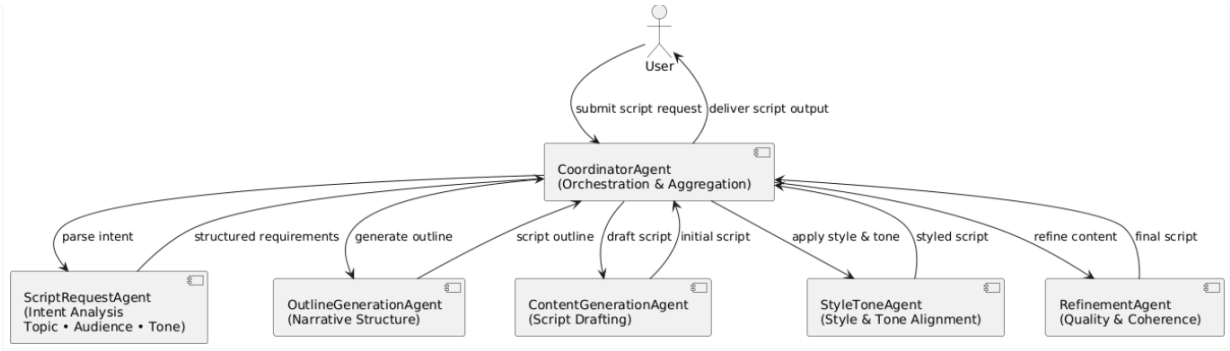

FIGURE 1: SYSTEM ARCHITECTURE DIAGRAM

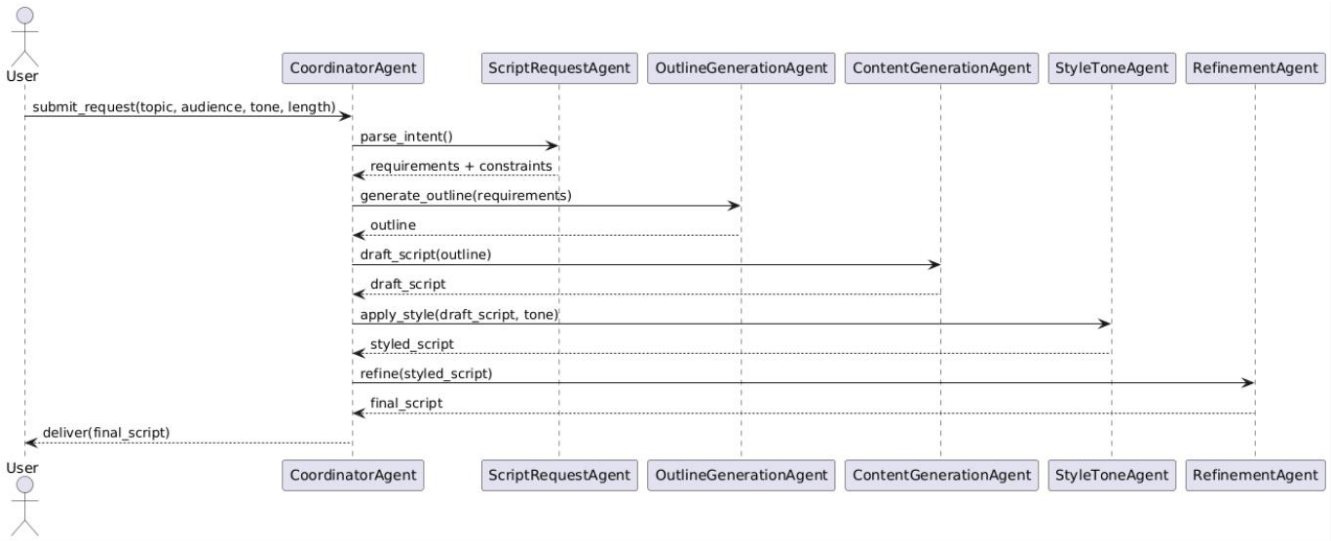

FIGURE 2: SEQUENCE DIAGRAM

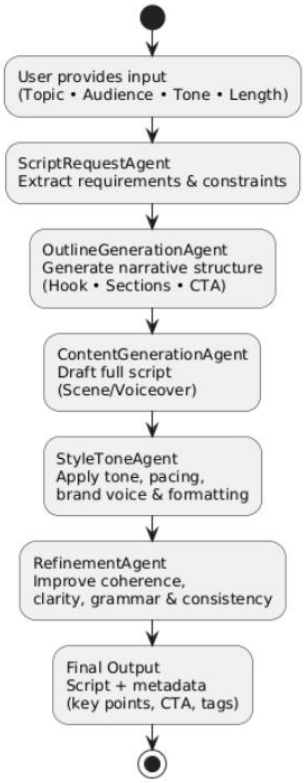

FIGURE 3: WORKFLOW DIAGRAM

1.3 Objectives

- To design and implement an AI-driven scriptwriting system based on a multi-agent architectural framework.
- To decompose the creative content generation process into clearly structured, interpretable, and modular stages.
- To demonstrate reproducible simulation and execution of agent-based text generation within a controlled experimental environment.
- To support and enhance human creativity by providing transparent, controllable, and assistive AI-generated content rather than fully autonomous generation.
- To enable context-aware script generation by incorporating user-defined constraints such as topic, audience, tone, and length.
- To ensure traceability and accountability in content generation by preserving intermediate agent outputs for inspection and refinement.

1.4 Impact

Business and Productivity Impact:

The proposed system significantly reduces the time and cognitive effort required to produce high-quality scripts, enabling faster content development cycles and improved productivity within creative teams. By automating routine aspects of script drafting, organizations can allocate human expertise to higher-level creative and strategic tasks, resulting in more efficient content production workflows.

Operational Impact:

The agent-based decomposition of the scriptwriting process promotes consistency in content structure, tone, and narrative flow. This structured approach minimizes the need for extensive manual editing and rework, leading to more predictable output quality and streamlined operational processes.

Ethical and Creative Considerations:

The system is designed to preserve meaningful human oversight by allowing users to review, inspect, and refine intermediate outputs generated by individual agents. This transparency supports responsible and accountable AI-assisted creativity, ensuring that final content remains aligned with human intent, ethical standards, and organizational values.

2. Selection of AI Technologies

2.1 Identification of Similar Systems

Commercial AI writing platforms such as Jasper, Copy.ai, and Grammarly GO provide automated content generation capabilities for marketing and professional writing tasks. These tools offer significant advantages in terms of speed, scalability, and ease of use; however, their internal workflows are typically opaque and tightly coupled to proprietary platforms. As a result, users have limited visibility into intermediate generation stages, restricting transparency, customization, and fine-grained control over content structure and style.

In academic and research contexts, automated text generation systems commonly employ end-to-end language modelling approaches that priorities linguistic fluency and coherence. While such methods demonstrate strong generative performance, they often lack explicit task decomposition and process-level explainability. This limitation makes it difficult to analyses how specific design decisions, prompts, or transformations contribute to the final output.

The proposed system builds upon these established approaches while addressing their shortcomings by emphasizing transparency, modularity, and user control through an agent-based architectural design. By explicitly separating the scriptwriting process into specialized agents, the

system enables inspection of intermediate outputs, supports iterative refinement, and enhances trust and accountability in AI-assisted content generation.

2.2 Innovative Approach

The originality of this project is demonstrated through its explicit adoption of a multi-agent architectural approach for creative text generation. Rather than relying on a single end-to-end generation process, each stage of the scriptwriting workflow is encapsulated within an autonomous and specialized agent. This design enables independent refinement of individual components and significantly enhances explainability by making intermediate creative decisions visible and inspectable.

Through coordinated interaction between agents, the system supports context-aware decision-making and produces more structured and coherent creative outputs. Intermediate results generated by earlier agents, such as intent interpretation or outline generation, directly inform subsequent stages of content drafting and stylistic refinement. Furthermore, the exclusive use of open-source tools and application programming interfaces (APIs) promotes accessibility, reproducibility, and ease of experimentation, ensuring that the system can be readily extended or adapted for both academic research and practical creative applications.

2.3 AI Methods and Tools Chosen

Table 1: Category Choice Justification for the AI Script Writer System

Category	Chosen Method / Tool	Justification
AI Paradigm	Agentic Artificial Intelligence	An agent-based paradigm enables decomposition of the scriptwriting task into specialized, interpretable stages, improving transparency and control compared to monolithic generation approaches.
Language Model	Large Language Model (LLM) via API	LLMs provide strong natural language generation capabilities required for coherent and context-aware scriptwriting across multiple domains and styles.
Task Decomposition	Multi-Agent Architecture	Separating intent analysis, outlining, content generation, styling, and refinement improves modularity, maintainability, and explainability of the creative process.

Prompt Strategy	Structured Prompt Engineering	Carefully designed prompts guide each agent's behavior, ensuring consistent output quality and alignment with user-defined constraints such as tone and audience.
Context Handling	Coordinator-Based Orchestration	A central Coordinator Agent manages agent execution order and aggregates outputs, enabling context-aware decision-making and controlled workflow execution.
Output Refinement	Iterative Agent-Based Refinement	Refinement and style agents allow systematic quality improvement, reducing grammatical errors, incoherence, and stylistic inconsistencies.
Development Framework	Jupyter Notebook	Supports interactive experimentation, transparent inspection of agent outputs, and reproducible simulation for academic evaluation.
Programming Language	Python	Python offers strong ecosystem support for API integration, prompt handling, and rapid prototyping of intelligent systems.
Technology Stack	Open-Source Libraries & APIs	The use of open-source tools ensures accessibility, reproducibility, and ease of future extension, aligning with academic and ethical AI principles.
Human Oversight	Human-in-the-Loop Design	User interaction at multiple stages ensures creative accountability and prevents fully autonomous, uncontrolled content generation.

3. Simulations & Development Setup

3.1 Simulation Environments and Setup Parameters

The system is developed and executed within a Jupyter Notebook environment that is fully compatible with Google Colab, enabling cloud-based execution without requiring local software installation. The implementation utilizes Python 3.x as the primary programming language, supported by relevant libraries for large language model API integration, prompt handling, and text processing. This environment facilitates interactive experimentation, transparent inspection of intermediate agent outputs, and reproducible simulation of the scriptwriting workflow.

3.2 Agent-Wise Task Distribution, Functionality, and Datasets

Each agent within the AI Script Writer system is assigned a clearly defined and specialized role within the overall scriptwriting pipeline, with explicit specifications for input data, processing responsibilities, and output artefacts. This structured allocation of responsibilities ensures that distinct stages of the creative process such as intent interpretation, outline construction, content drafting, stylistic alignment, and refinement are handled independently by dedicated agents.

Such modularity significantly enhances traceability, as intermediate outputs generated by each agent can be inspected to understand how the initial user input is progressively transformed into a complete and refined script. In addition, isolating functionality at the agent level simplifies debugging and validation, as any issues related to content quality, coherence, or stylistic consistency can be traced back to a specific agent without impacting the entire system. This design also supports systematic evaluation, enabling individual agents to be assessed, improved, or replaced independently while preserving the integrity and stability of the overall workflow.

Furthermore, the explicit definition of agent inputs and outputs promotes reproducibility and maintainability, making the system easier to extend with additional agents or adapt to alternative content generation tasks in future iterations.

TABLE 2: AGENT FUNCTIONS

Agent Name	Primary Function	Input Data	Output Data / Artefacts
ScriptRequestAgent	Analyses user intent and constraints	User input (topic, audience, tone, length)	Structured requirements and content constraints
OutlineGenerationAgent	Generates the narrative structure of the script	Structured requirements	Script outline (sections, flow, key points)
ContentGenerationAgent	Produces the initial script draft	Script outline	Full script draft (raw content)
StyleToneAgent	Applies stylistic and tonal adjustments	Script draft and style parameters	Styled and tone-aligned script
RefinementAgent	Improves quality, coherence, and clarity	Styled script	Refined final script

CoordinatorAgent	Orchestrates agent execution and aggregates outputs	User configuration and intermediate agent outputs	Consolidated final script and execution logs

3.3 Prompt Engineering and Data Processing

Prompt engineering techniques are systematically applied to guide the behavior and output of each agent within the AI Script Writer system. Carefully structured prompts are designed to reflect the specific responsibilities of individual agents, such as intent interpretation, outline generation, content drafting, stylistic adjustment, and refinement. This targeted prompt design ensures that generated outputs remain coherent, contextually appropriate, and aligned with user-defined constraints including topic, audience, tone, and length.

To support transparency and accountability, all intermediate prompts and corresponding agent responses are preserved throughout the execution process. This logging mechanism enables inspection of how each stage contributes to the final script and facilitates iterative refinement when output quality does not meet user expectations. By enabling systematic prompt revision and output comparison, the system supports continuous improvement of both agent behavior and overall content quality, while maintaining a clear audit trail of the creative decision-making process.

SECTION 4: Use Case Explanation with Evidence

4.1 Step-by-Step Workflow Execution

The workflow begins with a user request that specifies key script parameters, including the topic, target audience, and desired tone or style. The ScriptRequestAgent analyses this input to extract structured requirements and constraints, which are then forwarded to subsequent agents in the pipeline. Based on these requirements, the OutlineGenerationAgent constructs a coherent narrative structure that defines the overall flow and key sections of the script.

Following outline creation, the ContentGenerationAgent produces an initial script draft that adheres to the predefined structure. The draft is then processed by the StyleToneAgent, which applies stylistic and tonal adjustments to ensure alignment with the intended audience and communication goals. Finally, the RefinementAgent performs quality-focused revision, improving coherence, clarity, and linguistic consistency to generate the final script output.

2. Ingestion & Analysis Agent

One-liner: Cleans and normalises the raw draft, extracting topic, platform, tone, and key points into a structured state.

+ Code + Text

```
class IngestionAgent(Agent):
    # Basic ingestion agent. You can later extend this to use Gemini for richer key-point extraction.
    def run(self, state: Dict[str, Any]) -> Dict[str, Any]:
        script_input: ScriptInput = state["input"]

        # Simple placeholder key-point extraction (could be replaced by a Gemini call)
        key_points = [line.strip("-• ").strip()
                      for line in script_input.draft_script.splitlines()
                      if line.strip()]

        state["analysis"] = {
            "topic": script_input.topic,
            "platform": script_input.platform.lower(),
            "tone": script_input.tone,
            "key_points": key_points,
            "current_script": script_input.draft_script
        }
        return state
```

3. Hook Generator Agent

One-liner: Uses Gemini to generate multiple strong, scroll-stopping hooks tailored to topic, platform and tone.

```
class HookGeneratorAgent(Agent):
    def __init__(self, model):
        self.model = model

    def run(self, state: Dict[str, Any]) -> Dict[str, Any]:
        analysis = state["analysis"]
        topic = analysis["topic"]
        platform = analysis["platform"]
        tone = analysis["tone"]

        prompt = f'''You are an expert {platform} content hook writer.

Topic: {topic}
Platform: {platform}
Desired tone: {tone}

Generate 7 very strong, scroll-stopping hooks.
- Each hook max 15 words.
- No hashtags.
- Make them punchy and emotionally engaging.
Return them as a numbered list.
'''
```

HOOK GENERATOR AGENT

5. SEO Keyword Injection Agent

One-liner: Uses Gemini to generate SEO keywords, titles, description and tags based on the topic and platform.

```
class SEOKeywordAgent(Agent):
    def __init__(self, model):
        self.model = model

    def run(self, state: Dict[str, Any]) -> Dict[str, Any]:
        topic = state["analysis"]["topic"]
        platform = state["analysis"]["platform"]

        prompt = f'''You are an SEO specialist for {platform} content.

Topic: {topic}

1. Suggest 8-12 SEO keywords/phrases.
2. Suggest 3 video/post titles optimised for CTR.
3. Write a 2-3 paragraph description with those keywords naturally included.
4. Suggest 10-15 tags/hashtags.

Return JSON ONLY:

{{
  "keywords": ["...", "..."],
  "titles": ["...", "..."],
```

SEO KEYWORD AGENT

8. Thumbnail Caption Generator Agent

One-liner: Uses Gemini to generate short, bold thumbnail caption options to drive clicks.

```
class ThumbnailCaptionAgent(Agent):
    def __init__(self, model):
        self.model = model

    def run(self, state: Dict[str, Any]) -> Dict[str, Any]:
        topic = state["analysis"]["topic"]
        hook = state.get("structure", {}).get("hook", "")

        prompt = f'''You are a YouTube/Instagram thumbnail copy expert.

Topic: {topic}
Main hook: {hook}

Generate 10 ultra-short thumbnail texts:
- 2-5 words each
- Very bold and emotional
- No hashtags, no emojis
- Designed for high CTR

Return them as a JSON list of strings ONLY:
["...", "..."]
'''
```

THUMBNAIL CAPTION GENERATOR

```python
# Example usage demo
sample_input = ScriptInput(
    topic="How to add AI engineer related projects in RESUME",
    draft_script="""Want to make your resume stand out for AI Engineer roles?
    Add projects that clearly showcase your skills—like model training,
    data preprocessing, and real-world deployment. Highlight the tools you
    used such as Python, TensorFlow or PyTorch, APIs, and cloud or MLOps
    platforms. Keep each project crisp by stating the problem, the solution
    you built, and the measurable impact it created.
""",
    platform="YOUTUBE",
    tone="friendly, hype, beginner-friendly"
)

# Re-configure the model with the correct 'models/' prefix
# Listing available models to debug 'NotFound' error.
print("Listing available models:")
for m in genai.list_models():
    if "generateContent" in m.supported_generation_methods:
        print(m.name)

# If 'gemini-1.5-flash' is not in the list above, you might need to choose a different model.
MODEL_NAME = "models/gemini-2.5-flash" # Updated to an available model
model = genai.GenerativeModel(MODEL_NAME)

orchestrator = ScriptRewriterOrchestrator(model)
output = orchestrator.run(sample_input)

print("=== GENERATED HOOKS ===")
for i, h in enumerate(output.hooks, 1):
    print(f"{i}. {h}")
```

EXAMPLE: TEST CASE

```
=== GENERATED HOOKS ===
1. Here are 7 scroll-stopping hooks for your video:
2. Your resume needs AI projects NOW. Get noticed, get hired!
3. No experience? Easily add killer AI projects to your resume!
4. Unlock AI job offers! Your resume needs THESE project secrets.
5. Transform your resume from ignored to *interview magnet* with AI!
6. Don't mess up your AI resume! Add projects the RIGHT way.
7. Want to be an AI engineer? Make your resume future-proof and irresistible!
8. Boost your AI career FAST! Add these project gems to your resume.
=== SEO METADATA (titles) ===
- Land Your Dream AI Job: How to Showcase Your Projects on Your Resume!
- AI Engineer Resume Secrets: Make Your Machine Learning Projects Stand Out & Get Hired!
- STOP! Are You Adding AI Projects to Your Resume WRONG? (Expert Tips Inside)
=== FINAL READY-TO-RECORD SCRIPT ===
Hey future AI superstar!
Ready to land your dream AI job?
Then your resume needs AI projects NOW!
Seriously, they're how you get noticed, get hired, and absolutely crush the competition.

Look, the AI Engineer job market?
It's super competitive, no doubt about it.
Just listing your skills on a resume won't cut it anymore.
Hiring managers aren't looking for buzzwords; they want *proof* - proof that you can actually apply what you know.
And guess what?
That proof comes from impactful projects on your resume!
In this video, I'm going to show you exactly how to craft those killer AI projects that grab attention and land you interviews, even if you're just start

First up, let's talk about showcasing your *core AI skills*.
When you're picking out projects, think big!
You want to demonstrate the *full AI lifecycle*.
What does that mean?
It means showing off your awesome skills in every critical stage.
```

```
Remember, numbers aren't just good, they *speak louder than words* and prove your worth!

So, out of all these powerful tips, which one are you most hyped to implement on your resume first?
Drop a comment below and let me know!
I love hearing from you all.

If you're truly serious about crushing it and landing your dream AI Engineer role, then you absolutely can't miss out on more career-boosting advice like
Do yourself a favor: smash that like button on this video.
Subscribe to the channel if you haven't already.
And hit that notification bell so you're always in the loop and never miss an upload!
Let's get you hired!
=== THUMBNAIL CAPTION IDEAS ===
- AI Resume: Hired!
- Your Resume Needs AI.
- Get Noticed: Add AI!
- AI Projects = Job.
- Land AI Jobs NOW.
- Unlock AI Career.
- Stand Out with AI.
- Master AI Resumes.
- Future Needs AI.
- AI For Your Resume.
```

OUTPUT

4.2 Developed User Interface and External Interaction

The Jupyter Notebook interface provides an interactive environment that allows users to execute the scriptwriting pipeline in a step-by-step manner and review intermediate and final outputs produced by each agent. This interactive execution model enables users to monitor the progression of the script generation process, facilitating greater understanding of how individual agents contribute to the final output.

By allowing users to inspect, adjust, and re-run specific stages of the workflow, the interface supports meaningful creative oversight and iterative refinement. Users can modify input parameters such as topic, audience, or tone, and immediately observe the impact of these changes on generated content. This level of interactivity enhances user control, supports responsible AI-assisted creativity, and ensures that the final script remains aligned with user intent and quality expectations.

5. Conclusions

his coursework demonstrates the feasibility and effectiveness of adopting an agent-based architectural approach for AI-assisted scriptwriting. By decomposing the creative workflow into a set of specialized and autonomous agents, the proposed system enhances transparency, modularity, and user control when compared with traditional monolithic text generation solutions. This structured decomposition enables clearer inspection of intermediate creative decisions, improves explainability, and supports more accountable and responsible use of generative AI technologies.

The simulated evaluation indicates that the system has strong potential to support creative professionals by reducing the time and cognitive effort required for script development, while still preserving meaningful human oversight. The agent-based design enables iterative refinement and targeted quality control, allowing users to intervene at specific stages of the workflow rather than

relying solely on final outputs. This approach is particularly valuable in professional and academic contexts where consistency, clarity, and accountability are essential.

Future work could further enhance the system by incorporating multimodal script generation, allowing the integration of visual, audio, or scene-level cues alongside textual content. Additional extensions may include more sophisticated style adaptation mechanisms, integration with enterprise content management and publishing platforms, and support for real-time collaborative editing to facilitate multi-user creative workflows. Further evaluation across diverse script genres and user groups could also provide deeper insight into the system's robustness and practical applicability.

References

- Russell, S. and Norvig, P. (2021) *Artificial Intelligence: A Modern Approach*. 4th edn. Harlow: Pearson.
- Wooldridge, M. (2009) *An Introduction to Multi-Agent Systems*. 2nd edn. Chichester: Wiley.
- Jurafsky, D. and Martin, J.H. (2023) *Speech and Language Processing*. 3rd edn. Hoboken: Pearson.
- Brown, T. et al. (2020) 'Language models are few-shot learners', *Advances in Neural Information Processing Systems*, 33, pp. 1877–1901.
- Wei, J. et al. (2022) 'Chain-of-thought prompting elicits reasoning in large language models', *Advances in Neural Information Processing Systems*, 35, pp. 24824–24837.
- Liu, P. et al. (2023) 'Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing', *ACM Computing Surveys*, 55(9), pp. 1–35.
- Shneiderman, B. (2020) 'Human-centered artificial intelligence: Reliable, safe & trustworthy', *International Journal of Human–Computer Interaction*, 36(6), pp. 495–504.

- Floridi, L. et al. (2018) 'AI4People—An ethical framework for a good AI society', *Minds and Machines*, 28(4), pp. 689–707.