AI AND THE LIMITS OF NATURAL LANGUAGE FOR COMMAND

JAMIE FREESTONE

AUSTRALIAN ARMY RESEARCH CENTRE
/ Australian Army Occasional Paper No. 41

AI AND THE LIMITS OF NATURAL LANGUAGE FOR COMMAND

JAMIE FREESTONE

/ Australian Army Occasional Paper No. 41

SERVING THE NATION

ARMY.GOV.AU

© Commonwealth of Australia 2026

This publication is copyright. Apart from any fair dealing for the purpose of study, research, criticism or review (as permitted under the Copyright Act 1968), and with standard source credit included, no part may be reproduced by any process without written permission.

The views expressed in this publication are those of the author(s) and do not necessarily reflect the official policy or position of the Australian Army, the Department of Defence or the Australian Government.

ISSN (Print) 2653-0406

ISSN (Digital) 2653-0414

DOI: 10.61451/2675162

All enquiries regarding this publication should be forwarded to the Director of the Australian Army Research Centre.

To learn about the work of the Australian Army Research Centre visit https://researchcentre.army.gov.au

Cover image: Australian Army soldier inspects computer monitors at the Taji Military Complex, Iraq. Source: Defence Image Gallery. Photographer: CPL Oliver Carter.

/ CONTENTS

/ 1. INTRODUCTION

The world's militaries, like other large organisations, are currently grappling with how to use large language models (LLMs), which are a form of generative artificial intelligence (AI).[1] The US Army, for instance, aims to incorporate AI—including generative AI—at every level.[2] AI companies are already making sales to armed forces around the world, such as the United States Africa Command.[3] In Ukraine, Palantir supplied its Artificial Intelligence Platform (AIP) system (which includes an LLM interface) on a mobile app for use by personnel in warfighting; soldiers on the battlefield type in queries in natural language (NL) and receive real-time information curated by an LLM.[4] AIP also allows users to build *AI agents* (hereafter simply *agents*), another emerging AI technology predicted to be a multitrillion-dollar industry.[5] Agents are AI systems that are empowered to execute tasks on the user's behalf. Such agents would be prompted with NL and would then use an LLM to translate the user's prompts into a format that machines can understand, to write code to run programs, to generate text for emails and messages, or even to navigate and use websites on behalf of the user. No previous software has achieved this level of autonomy.

Such rapid adoption of untested AI raises many cybersecurity concerns.[6] LLMs evince bias in their output; their training arguably infringes on others' intellectual property; the models' inner workings are opaque even to the programmers, making it hard to predict fail modes in advance; and they are prone to so-called *hallucination*, whereby the text output of the LLM is untrue or nonsense.[7] The use of agents raises further questions as to whether current AI systems can or should act independently. Critics point out that the underlying models struggle with planning, continue to fail in some elementary reasoning tasks, and lack an understanding of the physical world.[8] There are also unanswered legal questions about who is responsible for the actions of an agent.[9] Furthermore, in a military and national security context, there are voices warning of the dangers of using AI for crucial intelligence analysis and making decisions on whether to go to war.[10]

Little attention has been paid, however, to how the capabilities of LLMs and agents are affected by their dependence on NL. This paper argues that *NL is the crucial bottleneck that will impede development, safety and reliability in this area, especially for agents*. Previously, computers were instructed by a programming language (PL) carefully designed for that purpose. NL, by contrast, is a natural phenomenon that evolved in unique circumstances to perform certain functions for bands of early humans. It did not evolve to command machines. Nevertheless, LLMs and agents are machines that, at least indirectly, are commanded by NL. This represents a paradigmatic shift in the history of computer science.[11]

The problem is especially poignant for the Australian Army. Modern military communications are supported by a well-honed body of protocols and training designed to overcome or sidestep the limitations of NL as a command language. NL is 'designed' for everyday human-to-human (H2H) communication, yet misunderstanding commonly occurs, sometimes with catastrophic consequences. Militaries do their best to standardise English or other NLs to reduce the risk of misunderstanding and deception. Some of the same measures can be employed to make LLMs and agents safer and more reliable. But, as explored later in this paper, even if NL-prompted systems are brought up to the same level of reliability as H2H communications, they remain unsuited to high-stakes tasks. While LLMs or agents can be used relatively safely in low-stakes tasks (such as summarising, querying and translating), this paper recommends against their use for tasks that involve high levels of trust or autonomous decision-making.

This study is interdisciplinary. The technology is emerging, and the associated risks are not fully known.[12] To address the risks posed by NL-prompted systems, it is necessary to range across the scholarly literature, from the evolution of animal communication and the philosophy of language to cybersecurity and the history of military miscommunication. The reader will be spared too many technical details, and many debates and open questions will have to be glossed over or relegated to endnotes. Instead, the paper presents sufficient information about language and AI to empower an interested reader to see the new class of risks entailed by this technology and the facts about language that make this technology different from other information technology, even other forms of AI.

The paper is arranged as follows. Section 2 looks at the different forms of communication covered here—including NL, PL, and command and control (C2)—and establishes a simple framework that allows for easy comparisons between them. Using this framework, Section 3 details how NL is different from PLs and why NL is therefore ill-suited to commanding machines. Section 4 elaborates on the problems that NL-prompted systems already face in this regard and details further problems that will limit their military utility in the future. Section 5 looks at ways to mitigate these problems and at the inevitable capability trade-offs. Section 6 lists recommendations for how the Australian Army should incorporate LLMs and agents into its operations and where their use should be restricted.

2. COMMUNICATION, COMMAND AND LANGUAGE

NL, PL, C2 and other systems of communication for humans, non-human animals, and machines are sufficiently different to warrant their own studies. Indeed, there are many disciplines that study them with widely varying methodologies and doctrines: linguistics, literary studies, the philosophy of language, communications, semiotics, biosemiotics, information theory, computer science, rhetoric, signal processing, ethology and so on. It is possible, though, to adopt a simple framework that foregrounds one feature of communication while remaining agnostic as to many of the particulars of communication debated in the literature. The central feature of this study is the notion of *command*—how to command a computer, how to issue commands in a military setting, and how difficult it is to command humans or machines using NL.

In this article, *communication* is defined as a sender's attempts to command a receiver or receivers. *Command,* in turn, is defined as a way to control another's behaviour without direct force.

These definitions may sound too broad or abstract. But they are based on work in the biology of communication in non-human organisms—which was, after all, the only form of communication for several billion years prior to the advent of both humans and machines. If we are to find the 'lowest common denominator' of communication—some characteristic that unites all forms of communication, no matter how modern or complex—it is in the relatively simple signalling systems that have evolved naturally.

Communication between predator and prey, between trees in a forest, between single-celled organisms, and even between cells within the same organism can all be understood as attempts to control behaviour by means other than direct force—for example, through the use of sound, visual display or chemical signals.[13] Implicit in this framing of communication is that there is a cost to communicating. It takes time and energy to send a message. From an evolutionary point of view, therefore, it is worth communicating only if doing so yields a benefit to the sender in excess of the cost. In the pitiless logic of natural selection, this benefit is more likely to be at the expense of the receiver. Most interactions, even among members of the same species, are competitive rather than cooperative.[14]

For cooperation to flourish, special conditions are needed. This typically means that the sender and receiver must be closely related, as in the case of bees who communicate pollen's location to their genetically similar sisters. Because of their common (genetic) interest, the rate of deception and ambiguity is very low. In contrast, the more common

examples of communication among non-human animals are in competitive scenarios, which are consequently marked by attempts at deception and by the use of highly ambiguous signals.[15] For instance, when a cat is threatened by another cat or a larger animal, it will make a hissing sound, its hair will stand on end, and it will arch its back. These audio and visual displays are forms of communication. They emphasise the cat's defensive abilities and make it appear larger to its opponent. The cat is 'lying' or at least exaggerating. The cat could only be disadvantaged by communicating an honest signal of its powers. It is in the cat's interest to deceive the opponent. Because this is a purely competitive situation, there is little 'honest' information in the cat's signals. The competitive versus cooperative nature of a communication system comes down to the evolutionary history of that system. Knowing the history is therefore crucial to predicting the amount of deception and ambiguity intrinsic to the system.

This brings us to the biologically unusual case of human communities. Here communication occurs liberally, especially via NL, amid a mix of cooperative and competitive incentives, producing a mix of honest and dishonest signals.[16] Unlike the bee dance, NL is frequently used to communicate with non-kin and even outright rivals, and so it is also used to deceive. And unlike the cat's threat display, NL is often used to share 'honest' information for the mutual benefit of the sender and receiver. We should therefore expect NL to be optimised for neither honest signals nor fake signals but for some blend of the two.

This is not the only difference between NL and communication in non-human animals. The signals in NL are far more complex than even the most sophisticated communication found among other animals, such as birdsong and whale song.[17] NL is *compositional*.[18] One can have a symbol *A* and a symbol *B* with different meanings, and these symbols can be combined to make *AB,* which has a third meaning that is not simply *A* followed by *B*. In other words, symbols can be combined according to a set of rules. This allows for an explosion of finely distinguished meanings. Indeed, other animal communication schemes are *closed* in the sense that new signals cannot simply be deployed ad hoc; they can only arise over time. NL is *open*, however, because a speaker of NL can produce utterly novel sentences and still be understood, provided the symbols being used are all known and are combined according to known rules.[19] Compositionality is one of the primary elements of grammar or syntax, whereby symbols are combined to compose extended or more subtle communiqués.

Another feature of NL that is not found elsewhere in nature is the heavy reliance on *context*.[20] Human conversation is not very explicit. Rather than specifying everything that a receiver needs to parse a sentence, the sender can outsource much of this to context. Context includes the immediate environment, perceivable by the speakers, in which the conversation takes place; the overlapping cultural context shared by speakers of the same language; and earlier parts of the discourse. For example, a sentence in an everyday

conversation might be, 'She didn't like those and we all know why.' Ripped from its context, the meaning is unclear. Pronouns like *she* allow speakers to save themselves constantly referring to people already named in the discourse. The first mention of someone might be *the woman who was staying in the hotel room next to mine*, but thereafter, one can say *she*. Similarly, a word like *those* may refer to some object currently in view of both speakers, and so its meaning is entirely dependent on current physical context; this is what linguists call *deixis*. Furthermore, the clause *we all know why* refers to some unspecified common knowledge, which may be highly culturally or personally contingent.[21] Context-dependency in NL means that two identical utterances can have totally different meanings if they are used in different contexts.

Ambiguity can thereby be resolved by context-dependency. This carries the benefit of efficiency, as one can reuse and recycle a smaller set of common words whose meanings are modified by contextual information. It also carries the cost of potential misunderstanding. But note that most human communication is not about *directly* commanding one's interlocutor; most conversation is about updating, remembering, gossiping and entertaining.[22] These functions of language might seem to bear little similarity to commands. But, returning to the evolutionary foundation discussed above, there is a cost to communicating that must be repaid. Sharing information for free, such as gossiping or improving a receiver's mood with a joke, compliment or reminiscence, is ultimately beneficial to the sender, too. These functions clearly aid social cohesion and help secure a mate, ally or friend.[23] Are they commands? In the sense used in this paper, yes. The sender is trying to control the receiver's internal behaviour and adjust their beliefs or dispositions in some way that will ultimately lead to an action—perhaps long delayed—that will benefit the sender. What might normally be called *informing*, *influencing* or *persuading* can be recast as delayed or deferred commanding.

Context-dependency is not needed for simple, non-compositional communication such as bird calls and chemical signalling. Nor is it strictly necessary for complex communication. For human-to-machine (H2M) communication, we employ PLs, which are compositional, have syntax, and allow for highly complex communication. Unlike NL, however, they do not rely on context or tacitly communicated information. For a computer program to run, the programmer must anticipate and make explicit all the information the computer will need to carry out the desired commands. Notably, ambiguity is also absent. Each symbol or function in a PL has one meaning only. PLs are deliberately designed and fit for purpose: absolute cooperation between the programmer and the machine. NLs, by contrast, evolved in conditions of mixed competition and cooperation among their users and hence contain considerable, but not total, ambiguity.[24]

Meanwhile, there are historical examples of how NL can be partially modified to make it a better, but imperfect, command language. Over centuries, militaries have developed C2 principles aimed at streamlining communication and reducing misunderstanding.[25]

Many standard practices in military communications are aimed at minimising ambiguity and making commands explicit: standardised vocabulary, formats for orders, pro-words, brevity codes, call signs, readback etc.[26] Without training and protocols, personnel would use untamed NL to communicate in high-stakes situations. Although it is not usually conceived of in these terms, C2 principles attempt to corral NL, which is not suited to giving commands, into a more formalised language, one that slightly resembles a high-level PL.[27]

Zooming out, we can consider the many different systems of communication embodied in nature and technology and see that, for our purposes, three important variables loom into view:

1. The complexity of senders' commands
2. The degree of ambiguity in those commands
3. The amount of contextual information required for receivers to follow the commands.

In light of these, we can briefly summarise the main forms of communication mentioned so far and how they appear from the point of view of communication as attempted command:

1. **Communication in non-human organisms.** Commands are simple and non-compositional. They are normally highly ambiguous because they are adversarial. No context is needed to 'decode' the commands.
2. **NL**. Commands are compositional and can be long and highly complex. Ambiguity is variable but ever-present because of mixed competition and cooperation. Context is necessary to resolve ambiguity and to complement the sparse information in the message.
3. **Military communications or C2**. Commands are compositional and can be highly complex. Ambiguity is reduced, but not eliminated, by training users to follow conventions and by the fact that users' interests are largely aligned. More context is supplied than for a typical NL but not as much as for a PL; context is still outsourced to users' overlapping cultural and institutional knowledge.
4. **PLs**. Commands can be very complex and compositional, necessitating rules of syntax or grammar. Ambiguity is minimal and the programmer and machine are 'fully cooperating'. Information is exhaustively provided so that a process can run without the need for extra context.

There are many other ways of analysing and classifying different communication systems. The categorisation proposed above does not invalidate any of them per se. It merely emphasises that even the more recent and more elaborate communication systems—

NL, PLs, C2—maintain the original, core function of primitive communication found in non-human organisms: to control another's behaviour without force—or, in a word, to command. Over time, other functions have been layered on top so that users of NL do more than simply and straightforwardly try to get another organism to flee. But even these newer functions are, at the bottom, about controlling the behaviour of one or more receivers in some way that will ultimately benefit the sender.

Given all this, the question becomes: *Is NL, with its various elaborations on simple commands, optimal for H2M communication*? Relatedly: *Can NL-prompted systems, like LLMs and agents, be comprehensively and safely commanded*?

This paper will answer both questions in the negative. To demonstrate why, consider that PLs *are* optimised for command in H2M. And yet PLs are quite different from the NL with which we command humans and, currently, LLMs and agents. So, what features are stripped from NL when designing a PL? Identifying those features can provide insight into the inherent drawbacks of NL as a command language and hence illuminate the drawbacks of NL-prompted systems.

The following section elaborates on the contrast between PLs (which certainly can work to command machines) and NL (which has yet to prove its utility).

3. THE WEAKNESS OF NL AS A COMMAND LANGUAGE

NL Versus PLs

There is surprisingly little scholarly work on the distinction between NL and PLs. There is a huge, polyvocal literature on the fundamental features of NL and a large and more settled literature on the workings of PLs; deliberate comparisons, however, are rare. Possibly this is because the differences are so stark as to seem obvious. Alternatively, it might reflect the lack of interaction between scholars of NL working in linguistics or philosophy and PL theorists in computer science. It is common to find the distinction mentioned in introductory programming courses, where instructors briefly explain how PLs are unlike the NL that novice programmers are used to.[28] The distinction is sometimes made in introductions to formal logic and linguistics too.[29] Given the relevance of the distinction between NL and PLs to this paper, it is worthwhile to outline it clearly.

One useful and widely read account of the differences is found in Pinker's popular work *The Language Instinct*.[30] He identifies five reasons why NL cannot be a formal language, of which PLs are a subset:

1. **Ambiguity**. Most words in English, for instance, have more than one meaning, which results in sentences with multiple interpretations. As examples, Pinker gives unintentionally humorous newspaper headlines: 'Iraqi Head Seeks Arms', 'Stiff Opposition Expected to Casketless Funeral Plan'.

2. **Lack of logical explicitness**. Consider the following statements: Ralph is an elephant; elephants live in Africa; elephants have tusks. A human reader will make the inference that Ralph lives in Africa and that Ralph has tusks. Strictly, these inferences cannot be made purely from the information supplied in the statements given in NL. We naturally supply the additional general knowledge that (a) the Africa Ralph is in *is* the same Africa mentioned in the statements and that (b) the tusks Ralph has *are not* the same as the tusks other elephants have. In a PL, earlier rounds of programming have defined the functions, arguments, variables etc., so that all necessary implications are specified somewhere in the stack.

3. **Co-reference**. This is the problem alluded to in the example above, whereby one can first refer to *the woman who was staying in the hotel room next to mine* and then, later in the discourse, refer to the same person as *she*. There need to be additional rules or knowledge to understand that the two expressions are equivalent in this context (the identical expression *she* will refer to other things in different contexts).

4. **Deixis**. Again, as mentioned above, words like *a*, *the*, *here*, *that*, *this*, *now*, *me* and *you* refer to things that are salient in the context of the current conversation.

5. **Synonymy**. There are many ways of saying the same thing. At the scale of words, there are synonyms that are virtually interchangeable: *serene* and *tranquil*; *tirade* and *diatribe*. At the scale of the sentence, it is easy to manipulate syntax without altering the meaning: the soldier fired a Javelin at the tank; the tank was fired at by the soldier's Javelin, etc. In a PL, there is often *some* leeway in altering the syntax to achieve the same result. But at the level of individual expressions, synonyms are simply redundant and are therefore not used. When one combines synonymy with ambiguity, as is the case in NL, it is possible to simultaneously have (a) many sentence constructions that mean the same thing and (b) identical constructions that mean many things. The only way to correctly parse them is with additional contextual cues—which is to say, information that is separate from the message.

Of these five reasons, two boil down to issues of context (deixis and lack of logical explicitness), and three boil down to matters of ambiguity (ambiguity, co-reference and synonymy).[31]

Beyond Pinker's classifications, the history of *natural language programming* is another useful (but oft-forgotten) basis for asserting insurmountable differences between NL and PLs. From the 1960s, there was a widely held ambition among programmers to make PLs increasingly user friendly and accessible. It was assumed that PLs would gradually come to resemble NL until one day it would be possible to program a computer without any coding experience, because the computer could be instructed using a PL indistinguishable from NL. NL programming did not eventuate. The reasons for its failure illustrate many of the problems with security and reliability that LLMs and agents already face.

The most common reason cited in this literature is, again, ambiguity.[32] Whereas PLs are designed to eliminate ambiguity entirely, NL is rife with it. The second most cited reason is the context-dependency of NL, although this exact phrase is not always used. Rather, commentators emphasise the difference between the procedural, explicit kinds of instructions entailed in computer programming, and the way things are expressed in NL.[33] The barrier to entry for programmers is not in learning the vocabulary of the PL (which might be fewer than a hundred expressions) but in learning to think like a programmer—that is, learning to specify or make explicit what the task involves and its success conditions:

> [P]rogramming is often defined as a process of transforming a mental plan that is in familiar terms into one that is compatible with the computer ... many difficulties arise because the distance between these is too large … There are also many ambiguities in natural language that are resolved by humans through shared context and cooperative conversation ... Novices attempt to enter into a humanlike discourse

> with the computer, but programming languages systematically violate human conversational maxims because the computer cannot infer from context or enter into a clarification dialog.[34]

Until the advent of LLMs, commanding a computer required a level of highly explicit language. This language was so far removed from everyday communication that achieving proficiency required a new way of thinking. This situation is not unique. Military C2 practices also require more explicit protocols than exist in everyday uses of NL. But in the case of C2, efforts to achieve common understanding benefit from the many shared assumptions of people working in the same NL who have the same (usually) national background and common training.[35]

Scholars of language might add further reasons why NL is not like a PL. In addition to ambiguity—where the same expression has multiple interpretations because of underspecified meaning—there are various kinds of *irony*. Irony involves expressions that have multiple intended meanings for multiple receivers. The simplest example is a sarcastic or snide comment: a naive listener will take the comment literally and miss the interpretation gleaned by a savvy listener. Another form of irony is the dog whistle used by a politician to signal one thing to a clique of supporters and another to mainstream audiences. An overlooked point in the literature on the evolution of NL is that a simple sender–receiver model misses the potential scenarios in which a speaker has multiple receivers with mixed agendas.[36]

The mere fact that this is possible in NL is a point against its unthinking use as a command language. Irony, along with euphemism and strategic ambiguity, is a way in which humans can tailor messages to audiences of several individuals who have non-overlapping interests.[37] An example from a military context would be the use of encrypted messages that mean one thing to an eavesdropper and another to the intended receiver. Fittingly, one of the best ways to see the differences between NL and PLs is to look at military communications.

Modifying NL in the Military

There are many modifications to NL that obviate the risk of miscommunication in the military. These systems somewhat resemble PLs in their reduced ambiguity and increased explicitness.[38] Somewhat tellingly, militaries inevitably face difficulties in their efforts to incorporate machines into these NL systems (H2M).

Miscommunication in the military is an important topic but has not attracted much scholarly attention. Nevertheless, there is strong institutional knowledge in the world's militaries of the cost of miscommunication and how to mitigate it. Canonical examples from history include the charge of the Light Brigade in the Crimean War, and the Battle of Gettysburg. Both involved vague orders that were misinterpreted, leading to disaster.[39] Techniques to

reduce ambiguity in commands include the introduction of phonetic alphabets, mandatory readback in audio communications, and standard formats for orders, such as the SMEAC (situation, mission, execution, administration/logistics, command/signal) format used by NATO and US forces.

These techniques reduce the inherent ambiguity in NL by standardising it for use in high-stakes military operations. The degree of standardisation is not as great as that found in a PL, but there are also PL-like systems found in C2.

Despite efforts to mitigate ambiguity, NL remains open to misinterpretation. A recent study by Schadd et al. concludes that 'Natural language is difficult for a computer to work with because it allows many variations in the representation of information'.[40] The same problems that afflict agents and LLMs are present in military systems, despite efforts to standardise communications between humans and machines. The fact is that NL remains so riddled with ambiguity and context-dependency that even in military systems streamlined for H2M there are still weaknesses. To overcome the challenges, militaries adopt principles to bridge the gap between what is said and what is needed to achieve the mission. One such principle is *command intent*:

> A common education, history, and mindset [i.e. shared context] may enlarge the set of intents that do not need to be mentioned. An example of an implicit intent is that taking casualties during a mission should be avoided. Differences in education and culture may cause differences in implicit intents between participants and may lead to undesired mission outcomes.[41]

There is no way for machines or humans to extract these implicit intents from the commands alone. They must already be known or be somehow embedded in the context surrounding the command.

The increased reliance on digital communication and information networks necessitates a battle management language (BML).[42] A BML is a way to communicate command intent on the battlefield and in simulations via some unambiguous, standardised language:

> [A] BML will not be a natural language … On the contrary, BML has to be a language whose expressions can be interpreted by systems. Thus, all those ambiguities and exceptions that characterize natural languages have to be avoided. Indeed, one of the central demands for a BML is its unambiguity.[43]

A BML is especially necessary between coalition partners with different NLs and different military cultures. Nowadays, such interoperability is needed for H2M and machine-to-machine communication (M2M) as well. Despite the best efforts of experts in the military, there is not yet a BML or any other language that can facilitate H2H communication *and* H2M or M2M communication at the same time. Anything based in NL is vitiated by NL's ambiguity and is therefore too imprecise to be interpreted and actioned by machines.

Under the definition of communication as a sender's attempts to *command* a receiver, there are clear parallels between managing orders in the military and instructing a computer. Countering the ambiguity and context-dependency of NL is already a priority in C2. It will also need to be a priority when incorporating LLMs and agents into the military. These systems will face problems in command previously applicable only to humans.

4. INHERENT PROBLEMS WITH NL-PROMPTED SYSTEMS

NL-Based Problems in LLMs

LLMs and agents already evince weaknesses peculiar to NL-prompted systems. Like humans, who also receive commands in NL, LLMs and agents can simply be fed disinformation in NL form. For H2H communication in the military, the threat of disinformation is ever-present. For example, an adversary may introduce false commands into a radio network. Much as a human may be taken in by a well-crafted message in NL, NL-prompted systems can be lied to, manipulated or confused by malicious prompts. Cybersecurity experts have raised alarms about these vulnerabilities, including the threat of *jailbreaking*: the process by which 'attacks are engineered to elicit behaviour, such as producing harmful content or leaking personally identifiable information, that the model was trained to avoid'.[44] The would-be jailbreaker can use various prompting techniques to circumvent the model's defences. These include asking the model to play a role or character, using appeals to logic or other persuasive techniques, tricking the model into entering developer mode to bypass the usual safeguards, and paraphrasing requests in unusual or coded language.[45] These attacks are unique to NL-prompted systems.

Most worryingly, NLs are vulnerable to *prompt injection* (PI). The threat of PI stems from the fact that a user can enter anything into the prompt window. This has been known since 2022, early in the development of LLMs:

> Due to the unstructured and open-ended aspect of GPT-3 prompts, protecting applications from these attacks can be very challenging. We define the action of inserting malicious text with the goal of misaligning an LLM as prompt injection.[46]

There is disagreement in the literature as to whether PI is a sub-type of jailbreaking or the other way around.[47] Regardless, there is some consensus regarding why PI is so dangerous, namely that 'LLM-Integrated applications blur the line between data and instructions'.[48] In a PL, this distinction is absolutely clear. One can enter *instructions* in a given PL and the computer will execute them, provided they are well formed. Anything else is ignored by the computer. If one wants to provide *data*—such as a file or object of some kind—it can be referred to in the PL but cannot be directly entered into the coding interface. In LLMs, by contrast, instructions, data, questions, gibberish, code, comments, malicious requests, or any other permutation of keystrokes can be entered into the same prompt window. Regardless, the model will generate text in response to *any* input. This is a radical break from any previous software.[49]

A related vulnerability is indirect or *remote* PI. The lack of separation between data and instruction carries over into the data that is retrieved by the model (for example, data collected in a web search or data retrieved when a model is given access to external memory). This allows adversaries to 'remotely affect other users' systems by strategically injecting the prompts into data likely to be retrieved at inference time'.[50] Note that *inference time* is when the model is responding to a prompt from a user. *Inference time* can be contrasted with *training time*, which occurs before the model is deployed and when it is pre-trained on a large dataset and fine-tuned to improve performance. Remote PI can:

> lead to full compromise of the model at inference time … This can entail remote control of the model, persistent compromise, theft of data, and denial of service. Furthermore, advanced AI systems add new layers of threat: Their capabilities to adapt to minimal instructions and autonomously advance the attacker's goals make them a potent tool for adversaries to achieve, e.g., disinformation dissemination and user manipulation.[51]

Unlike direct PI, which requires the adversary to have access to the model's prompt window, remote PI means they can plant adversarial prompts somewhere they know the model will access. Effectively, they lay traps that will be unwittingly activated by legitimate users.

A related type of attack is called *data poisoning*. In data poisoning, adversaries affect the model earlier in the process by injecting adversarial prompts into data which they suspect will be included in the model's pre-training phase. This includes methods as simple as editing publicly available web content like Wikipedia.[52] If retrieved and ingested, this data can indirectly control the model or lead the model to violate its usual safeguards.[53]

Remote PI and data poisoning expand the scope of NL-based vulnerabilities in LLMs and agents. Like direct PI, they succeed because of the blurring of data and instructions. During training, a model learns statistical associations between data, mainly comprising NL text. When it comes to inference time, it is prompted with an instruction that is also in NL. An LLM's basic function is to predict and output the next most likely word, given the words in the prompt. If the prompt (instruction) mimics a pattern found during training (data), the LLM will complete that pattern. This is what enables LLMs to respond to prompts in useful and relevant ways. Their instructions are in the same format as their data because they have been built to essentially produce more of the data they were trained on—that is, more plausible stretches of natural language. This functionality can be exploited either by a cleverly chosen prompt (PI), by having the LLM access a deliberately prepared external site (remote PI) or by curating data on which the model will be trained (data poisoning).

NL-Based Problems in Agents

The problems outlined above apply to any agents that are operated via a prompting window. One might think that commanding an agent is therefore just as fraught as commanding an LLM, but agents are more concerning still. Unlike LLMs, which are usually restricted to generating text or other media such as images, an agent's *outputs* will be far more diverse.

Agents are a new technology, and most of what has been written about them is speculative.[54] Media coverage of Big Tech's proposed agents extols the possible use cases and probably exaggerates the size of the market.[55] Common examples of use cases for a consumer-level agent include organising a holiday, making purchases and booking restaurants.[56] Over the last two years, agents have failed to attain anywhere near the reliability required for consumer use. Agents have been successfully deployed, however, 'in the background'. Salesforce, for example, provides agents to private enterprise; these are set up to perform highly automatable tasks in a business context.[57] This is quite different from the general-purpose concierge or butler envisioned by AI enthusiasts. The important distinction is that task-specific agents will be carefully tested, calibrated and instructed with highly structured prompts and orchestration layers. By contrast, general-purpose butler-style agents will be prompted in unconstrained NL and will be expected to perform their tasks with minimal feedback and instruction. For these general-purpose agents, the limitations of NL will be amplified in several ways.

First, agents inherit the NL-related vulnerabilities of LLMs. This includes an inability to resolve ambiguities in user prompts, a vulnerability to PI and jailbreaking, and a lack of the context required to parse many NL messages.[58]

Second, agents may be embedded in ecosystems of other agents.[59] Multi-agent systems, with autonomous agents communicating among one another in NL, may become vectors for PI attacks. A malicious prompt can self-replicate across interconnected agents, behaving much like a computer virus spreading through their NL communication. This phenomenon, dubbed *prompt infection*, has been shown to pose severe threats, including data theft, scams, misinformation and system-wide disruption.[60] Again, the threat of NL being used as a de facto PL is recognised by some on the cutting edge of cybersecurity research. Commenting on the prospect of agents interacting in a web of imprecise NL communication, one recent study concludes, 'the imprecision and ambiguity of natural language could also inadvertently pose safety risks, even in the absence of an adversary'.[61]

Third, and most worryingly, agents will also use other systems, thereby expanding the footprint of any NL-based vulnerabilities. For example, humans will communicate to agents in NL and then these NL commands will not only produce more content (similar to LLMs) but will also translate to an open-ended set of real-world actions.

Issues Related to Commands

By contrast to LLMs, agents can be linked to external applications, can instruct other agents or LLMs, and can be given control of a user's computer.[62] In effect, agents will either translate NL commands into a PL (to initiate some action) or receive NL commands and then translate them into further commands in NL. Either way, these linkages introduce further risks associated with ambiguity in language and context-dependency.

There are at least four methods by which agents will take commands given in NL and convert them to actions either for M2M or M2H:

1. **Writing code**. LLMs have become invaluable 'copilots' for software engineers. Not only can an LLM auto-complete likely stretches of code; it can also be prompted to create code in a given PL in order to do 'X'. Normally, the programmer needs to enter a process of back-and-forth or trial and error to refine the code generated by the LLM. This process, though imperfect, can still boost a programmer's productivity.[63] Less useful is so-called 'one-shot' code generation, where LLMs are provided with a single example of code and asked to extrapolate and produce something similar for a new purpose. This is a time-saving activity that comes with trade-offs: 'complex tasks requiring detailed understanding and context may pose challenges for one-shot prompting models'.[64] Agents, however, will be empowered to write code and attempt it on a first try. This represents a straightforward case of the user instructing the agent in NL and then the agent trying to shoehorn an ambiguous NL command (which may lack explicit context) into something explicit and unambiguous in order to command another system. But command intent may be lost in the translation.

2. **Application programming interfaces (APIs)**. Agents can call APIs. APIs are the main way in which two software applications communicate to each other. They are part of the backbone of modern ICT infrastructure, with many billions of calls made per day and trillions made every year. For example, when using a rideshare app, the mapping information might be provided by a second application (mapping software) which the rideshare app *calls*. For the call to work, the rideshare app must submit some information in a structured format to the mapping software. This will commonly be in a format such as a JSON schema or an XML document. These are simplified formats that are readable by humans and machines; they contain numbers and natural language but are laid out under predefined subheadings and categories. With the mapping example, the API might require the rideshare app to include information such as a request for the user's current location, a request for the destination, a request for the fastest route between the two, etc. This allows the mapping software to deliver the information required. All of these actions are performed in an M2M communication pipeline. For this reason, constrained,

unambiguous and necessary context needs to be provided. To populate the API call, the agent must take the user's NL prompt and convert or translate that into words or numbers that fit the predesignated categories. The API then commands another machine (the third-party software) to do something. Given the indirect method of communication and the imperfect translation of NL into PL, there is a risk that command intent is lost.

3. **Computer use (CU)**. One way to bypass the permissions and access issues raised by third-party apps is to empower the agent to use one's computer. Using a multimodal model, the *CU agent* can operate the user's web browser, take screenshots, interpret what text or other fields are available, and then click on links, check boxes or enter text.[65] Effectively, this is another case of giving a machine an NL command—'book me flight to Melbourne next Wednesday, some time around 7, economy'—and expecting it to sift through the ambiguity and lack of context and carry out the user's intention. In this example, how close is 'close' to 7 pm? Is 'next Wednesday' the very next Wednesday or Wednesday of next week? What details will the agent need that are missing from the prompt? Is the agent empowered to guess at these, or will it need to enter a back-and-forth with the user? CU comprises a mix of communication styles. Given any particular NL prompt, a CU agent effectively converts that NL into PL by taking certain actions, like clicking buttons to navigate a website. (It also involves translating NL to NL; some of its actions might be entering NL text.) Because a CU agent can initiate any downstream commands that are accessible through a computer's software, these actions can have arbitrarily large real-world consequences.

4. **M2H**. Agents can be incorporated into organisations as 'team members'. Already, some businesses have hybrid workforces comprising humans and agents. Agents may be included on a company's communication platform (e.g. Slack) so that human workers can message the agents. It is envisaged that many more tasks will be delegated to agents and that many white-collar workplaces will include hybrid teams revolving around H2M and M2H communications.[66] This will involve agents—who were initially commanded in NL—potentially commanding humans in NL. This invokes all the language-based problems associated with LLMs, such as PI, jailbreaking and hallucination.

Issues Related to Actions

The potential range of actions open to agents is vast. Their level of autonomy is greater than that of both normal software and standard LLMs. For NL-prompted agents, their autonomous actions will be downstream of commands given by humans, or perhaps other agents, in NL. In one way or another, they will need to translate NL (with its ambiguity and context-dependency) into a more structured PL or semi-formal language. This process is imperfect, so the transmission of command intent cannot be guaranteed.

To illustrate, the following are examples of actions that will be instigated by one or more of the above communication channels, all of which are susceptible to problems associated with NL.

1. **Tools and apps**. There are already many applications that utilise LLMs and are therefore vulnerable to PI-like attacks.[67] The ecosystem of linked apps, plugins, web tools and other software that builds onto an LLM's functionality is growing rapidly. Existing LLMs tend to engage third-party apps or external knowledge bases only on the user's request or under the guise of a particular solution, such as an LLM performing a web search. An already standard method is retrieval-augmented generation (RAG), whereby an LLM makes an API call to retrieve some additional documents, reads the text and then augments the user's prompt with what it found, to then generate a response. Agents are envisioned to use these tools more readily and more autonomously. This will mainly occur via API calls but could also include writing and executing code.

2. **Robots and other autonomous systems**. Researchers have been quick to explore the possibility of pairing the powerful new input of an LLM with a new generation of robots.[68] Typically this is an onboard LLM so that a robot can be instructed by a human in NL. But some have mooted the possibility of agents having remote control of robots, either by calling the robot's API or by generating code.[69] This logic could extend to the command of other autonomous systems, such as autonomous vehicles or control systems. In a military setting, that might involve a human instructing an agent who then carries out orders; these orders could include controlling autonomous systems such as vehicles, sensors and weapons systems. Again, this would probably involve something like an API call or other structured protocol—but the initial prompt would be in NL, which the agent would then translate into subsequent commands.

3. **The user's own device**. Agents on a desktop computer or portable device can be asked to perform a task, which they then complete by operating the user's software. The user temporarily cedes control of their device to the CU agent. This involves giving an agent some or all of the permissions of the human user. In principle, the CU agent could perform any action on the computer that a human user could, save perhaps for actions that require biometric ID (fingerprint recognition etc.). This includes composing text and communicating in NL to other agents or humans: M2H. Clearly, CU opens opportunities for hackers. What is more germane to this study is that CU also allows the agent to use NL in yet another environment where humans already do so in a manner that is far from foolproof.

When considering communication as a way to command receivers, the vulnerabilities identified above are particularly worrying. Traditionally, 'normal software' does precisely

what it is commanded to do. The system is protected against misunderstanding and jailbreaking by the fact that commands must follow rigid protocols (they must be well formed according to the syntax of the PL) and only trained operators who know these protocols can command them. By contrast, agents combine the solicitude of normal computer systems with the potential for deception inherent in language. They have the same NL interface as a human interlocutor, but with none of the natural defences against deception that humans possess.[70]

Interim Summary

Given the issues raised above, it is worthwhile to reconsider the questions raised in Section 2: *Is NL, with its various elaborations on simple commands, optimal for H2M communication*? Relatedly: *Can NL-prompted systems, like LLMs and agents, be comprehensively and safely commanded*?

The answer to the first question must be *no*. Given the evolution of NL, the history of PLs and the already apparent vulnerabilities of LLMs and agents, it seems clear that while astonishing advances in LLMs mean that machines *can* now be commanded with NL, this does not mean they *should* be. Concerningly, this reality is largely being overlooked in existing literature, probably because the topic engages with issues of an interdisciplinary nature. Scholars in the evolution of NL, for example, may have little interest in LLMs and agents or, if they do, they may not yet have been able to publish work in this new field. The author's personal communication with leading scholars in the field confirms, however, that the ambiguity and context-dependency of NL are features that are 'baked in' for evolutionary reasons and this makes NL irredeemably different from a PL.[71]

As to the second question, experts in cybersecurity have already flagged vulnerabilities in LLMs and agents, some of which (like PI and jailbreaking) are attributed to inherent problems with NL commands. These researchers, meanwhile, are understandably unaware of work in linguistics, animal communication and the philosophy of language that would reinforce their concerns. But while NL may not be the *optimal* way to command machines, its evolutionary history also means it is an accessible and flexible way for humans to perform H2M on a range of tasks. NL can probably never be made totally safe because it cannot be made totally unambiguous and explicit. The relative safety of this method of command will therefore depend on the level of risk entailed in a given task and the mitigation strategies used to somewhat constrain or formalise NL.

Once again, there is an historical irony here for militaries. There are already many ways in which NL has had to be constrained or modified to counter its weaknesses as a command language in a military context. The following section looks at how to extend the logic of these modifications (detailed in Section 3) to enable LLMs and agents to be cautiously incorporated into the Australian Army.

5. HOW TO MITIGATE H2M RISKS IN THE AUSTRALIAN ARMY

The risks peculiar to NL-prompted systems—PI and jailbreaking—are a new field of study without a clear scholarly consensus. Because they are NL based, they are in some respects more akin to the way H2H communication can be hijacked than to traditional attacks in computer security. Some experts argue that these problems are insurmountable and demonstrate a language model's inherent vulnerability.[72] It remains to be seen if this pessimistic conclusion holds. Certainly, no model is yet beyond the reach of jailbreaks. In fact, there are many online competitions to see who can be first to jailbreak a new model upon release, and the competitions are invariably won within hours.[73] But there are precautions that can be taken to deal with the problems associated with NL.

To mitigate H2M risks, especially those arising from the ambiguity of NL, the Australian Army can draw on existing strategies from both PLs and C2 practices. These strategies fall into two broad categories: communication protocols and training. Protocols shape the system of communication, while training shapes the users of that system. The strategies will be familiar already to those versed in military communications.

None of the measures outlined below are perfect. They all involve trade-offs. And while they may reduce the ambiguity and increase the explicitness of H2M commands, they cannot eliminate ambiguity altogether or capture all relevant context. Importantly, given the high-risk context within which military operations are conducted, the strategies need to be employed conservatively. Unlike humans, machines remain impervious to the cultural and psychological factors that can make H2H communications trustworthy, even accounting for the ambiguity that permeates NL.

Protocols

One mitigation strategy is to constrain model inputs to structured prompts only.[74] This reduces ambiguity but undermines a key strength of LLMs and agents: their accessibility and flexibility. Users benefit from the functionality of LLMs and agents precisely because they can interact in plain language without knowing a formal syntax or PL. Constraining prompts to a rigid schema, such as JSON, imposes a trade-off: flexibility and accessibility in return for precision and security. And even then, security is not totally assured. Jailbreaking may become harder but can still occur even with restricted prompt formats.[75]

Requiring the model to *back brief* the user is another way to reduce ambiguity through the enforcement of a protocol that helps ensure that the user's intent will be carried out.[76] Back briefing is commonly used in business communication and military communication. It can be thought of as a higher-level version of readback. Unlike simple readback (which merely duplicates the message to reduce ambiguity), back briefing involves the system paraphrasing or summarising the user's intent to confirm that the right action will result. Of course, this process cannot ensure that an agent will actually carry out the action intended by the user or, indeed, that it will act at all. But it can improve confidence in the relative unambiguity of command that has been transmitted in NL. Some LLM-powered systems already use a variant of this technique by asking clarifying questions to improve results.[77]

Again, the trade-off of back briefing is a loss of flexibility. Speed of communication is also compromised. More back briefing and clarifying means slower results and potentially a narrower range of queries or requests that are accepted. Further, some legitimate requests may be denied or rerouted along with the bad or illegitimate requests. Back briefing is nonetheless preferable in many circumstances to enforcing highly structured prompts. In fact, as users interact with a system that asks clarifying questions, they will (in effect) be informally trained to give more precise and context-inclusive prompts.

There is one crucial element missing from H2M protocols that arguably explains much of the success of H2H communication in the Army and elsewhere: *trust*. Much has been written already about trustworthy AI and autonomous systems.[78] Often this discussion involves an anthropomorphic notion of trust, an idea that can be contrasted with the feeling of 'trust' one has in a machine that it is performing its function as intended (i.e. *reliably*).

At least some of the trust that humans place in their human interlocutors is owing to a network of obligations, promises, punishment, guilt and sympathies. Humans are trustworthy in their communication partly because other humans recognise the incentives that dissuade them from lying. Humans are subject to punishment and feel negative emotions when they lose face or are exposed as liars. This trust network forms a set of protocols that partly enforces honest communication. AI and autonomous systems are, as yet, not amenable to these emotional safeguards.[79] Apart from any question relating to the linguistic competence of LLMs and agents, or their general reliability as machines performing functions, they certainly cannot be included in any trust network—to the extent that trust is partly founded in emotional and psychological notions so far unique to humans. In the military, for instance, they cannot be integrated into a rank architecture for the same reasons they cannot be court martialled. Because they cannot be included in protocols of trust, they cannot be included in C2 systems in the same way as human communicators.

Training

Ambiguity can be mitigated by formally training the users of a system. Training increases shared context and promotes unambiguous phrasing. Teaching personnel to only submit prompts that are properly formatted is certainly a good way to lower the risk of PI and similar jailbreaks. But, as researchers in the field recognise, the more secure the model is, the less accessible it is, and vice versa.[80] For example, prompt formats can be enforced, either by conforming to prompt engineering standards—such as the CO-STAR framework[81]—or by requiring users to input only highly structured prompts like JSON schemas. This is a worthwhile exercise for developers who are trying to get high performance from models on complex tasks. For ordinary users, however, such requirements introduce complexities to the process that are a barrier to entry. Indeed, they introduce standards akin to requiring users to learn a PL to interact with the software.

Another approach to training would be to introduce new accountability structures. If personnel are trained to be responsible for misunderstandings, this will channel human operators to verify a system's output before executing orders, for example. This is concordant with the many existing recommendations for retaining human oversight and accountability for AI decision-making systems.[82]

The other side of the equation involves training the non-human users—that is, the LLMs or agents. An agent can be fine-tuned: trained on tasks and data specific to an organisation. For example, an agent that is fine-tuned on military tasks, jargon and doctrine would be more likely to interpret a user's commands correctly and more likely to issue commands that are intelligible to personnel and that align with institutional norms. Such customisation, however, involves a trade-off with security. The more customised and context-aware an agent becomes, the more data it must access, including sensitive data. This is sometimes called the *personalisation–privacy paradox* (PPP), wherein consumers often prefer personalised information technology but must yield—knowingly or not—private data to access the benefits.[83] The PPP extends to new AI systems such as agents.[84]

In a military setting, personalisation down to the level of individual users would be both unlikely and insecure because individual user data would need to collected and stored. A more likely approach would be to grant persons of different rank or clearance distinct levels of access to data and control over agents or LLMs. Someone with higher clearance would, for example, be able to use prompts that utilise RAG for access to more sensitive data. This moderate customisation would require access controls or credentials, like any secure software. This would be a way for models to be customised for a *class* of user rather than for an individual user.

Further personalisation from agents and LLMs will come from models that can learn about their individual users. Today's models do virtually all their learning during their initial

training runs and fine-tuning phases. This enables them to adapt to the information within a given session or context window and thereby achieve some *in-context learning*.[85] For example, during a session, a human user might stipulate meanings for certain terms to avoid ambiguity. Once the session ends and the context window is closed, however, the model will reset and the custom terminology is forgotten. Further personalisation will be possible if *continual learning* (also called lifelong learning or online learning[86]) becomes available. Continual learning would empower a model to learn from successive interactions with users, effectively making its everyday operations part of its training.

Continual learning is not currently possible, although AI companies see it as one of the most important and lucrative future developments.[87] Such models would be most able to avoid misunderstanding by learning an individual user's preferences and idiolect over repeated and remembered interactions. This capability would substantially improve H2M communication and reduce the risk of misunderstanding by making the agents privy to otherwise unavailable context and tacit assumptions from particular users. In other words, they would emulate at least some of the processes humans go through when learning to communicate with close allies. While it offers potential for enhanced H2M communication, continual learning also introduces new risks associated with the model's memory—so-called *catastrophic forgetting*. This term refers to a situation whereby 'old tasks or data are rapidly forgotten as the training distribution changes'.[88]

Alternatively, a model could be made to learn less. By freezing a model such that it will not even engage in in-context learning, the predictability of its outputs can be increased. The flip side is that its flexibility and accessibility would be hamstrung.

6. RECOMMENDATIONS FOR ARMY

Assumptions About Risks and Threats

In the Australian Defence Force, a complete risk assessment for a new technology would generally examine both the technologies intended to be introduced and the systems in which they will be embedded.[89] It would also consider risks against five levels of consequences (insignificant, minor, moderate, major and catastrophic).[90] It is beyond the scope of this paper to make such a detailed assessment. Instead, this paper groups the risks against the measurement of *high stakes* versus *low stakes*. *Low stakes* corresponds to insignificant and minor consequences. *High stakes* includes moderate, major and catastrophic consequences.

While the threat landscape is broad and rapidly evolving, this paper has demonstrated that NL-prompted systems are inherently vulnerable. Based on the existence of the multiple attack vectors outlined (e.g. jailbreaking, PI, remote PI, data poisoning), *it must be assumed that these systems will be compromised at some point*. Further, the unreliable nature of NL-prompted systems means that *accidental misuse or negligent actions are also certain to occur, occasionally*. The recommendations below are therefore aimed at limiting the risks, inevitable as they are, to low-stakes uses and avoiding high-stakes uses altogether.

NL is suboptimal as a command language but has been tolerated for H2H communication for thousands of years. In contemporary high-stakes military contexts, NL is modified to reduce ambiguity. Many lessons have already been learned from H2H communication about the risks of miscommunication with machines. These lessons inform the recommendations outlined below. These measures should be seen as an attempt to *over*-compensate for NL's ambiguity and context-dependency. Merely *equalling* the level of risk tolerated in H2H would be problematic.

In making the recommendations, it is acknowledged that there are other risks associated with LLMs and agents. These include factors such as bias and model drift, which are not the focus of this discussion. These recommendations should not be taken as applying to new technology in general. For example, we know that 'Army aspires to leverage emerging technology such as Artificial Intelligence, autonomy, and robotics to gain operational advantage',[91] but this paper does not purport to assess the risks involved in other AI paradigms. The recommendations cover NL-prompted systems only.

List of Recommendations

- **Restrict LLMs and agents to low-stakes, non-operational tasks**. These include summarising documents; translation; querying internal databases; and other administrative, research or office tasks. Errors and miscommunications in these domains are relatively low-consequence, at least in terms of their first-order effects.
- **Maintain strict permissions for LLMs and agents that are linked to other systems**. Where NL-prompted systems will engage other systems (through linked tools, RAG, or control of autonomous systems), strict permissions should be in place to limit the actions they can trigger or the information they can access. For example, roles-based access control (RBAC) is a standard way to make sure that different users of the same system are permitted to access information commensurate to their rank or clearance. Ensuring these systems adhere to RBAC, as with any other software, is essential.
- **Encourage low-stakes use**. LLMs continue to improve. It is crucial that the Australian Army continues to be a modern work environment committed to operational excellence; therefore, the productivity and capability gains of current and near-future systems are too great to ignore. In low-stakes areas, Army should continue to follow guidelines that ensure a high standard of accountability and prudence, in line with other federal government standards—for example, the Commonwealth Ombudsman's better practice guide for automated decision-making.[92]
- **Enlist LLMs in cybersecurity where appropriate**. The defensive capabilities of LLMs should not be neglected. LLMs can analyse, summarise and translate NL in a manner superior to humans (at least in terms of speed, volume and versatility, if not accuracy or trustworthiness). They can be a valuable ally in processing large amounts of text, including screening for likely jailbreaking and PI attacks. For the many reasons outlined above, they do not offer a 'silver bullet' but, ironically, they can be part of an evolving cybersecurity solution.
- **Avoid the use of agents in tasks requiring direct action or autonomous decision-making in open-ended environments**. Where agents are empowered to act or make decisions, the space of possible actions or decisions should be tightly circumscribed. The lack of strictly defined input makes the output of such systems unreliable. For example, an agent might be empowered to triage information (i.e. flag it as important or unimportant). While the agent's decision-making might be compromised, in this example the potential consequences are limited to either missing important information or amplifying unimportant information. Provided these risks are known and tolerated, there is no scope for the agent to do anything beyond this task.

- **Do not allow agents to issue orders or interact directly with C2 systems**. The problems of commanding agents with NL are great enough without requiring them to *issue* commands. Agents are NL users, but they are not responsible in the same way that human NL users are. They cannot be influenced by incentives or held to account by peers. Hence, they cannot be responsible for mistakes. An exception could be made in simulations or tightly controlled test environments.
- **Where practical, enforce structured prompt formats**. Use frameworks like CO-STAR or controlled NL schemas to reduce ambiguity when working with LLMs and agents. This may be more effort than it is worth for some applications.
- **Embed back briefing or clarification loops into NL-prompted systems**. This is especially important for any agent tasked with planning, scheduling or data analysis. Treat this as a minimum safeguard against misinterpretation. It improves user performance and reduces model risk. While back briefing can clarify the intent of a command, it cannot ensure the command is actually carried out.
- **Train personnel to use NL-prompted systems**. Training could include instruction on structured prompting, common failure modes, adversarial attacks (e.g. PI), and red-teaming exercises to expose the potential for misuse.
- **Fine-tune models only on domain-specific, institutionally vetted data**. Doing so will lower the risk of data poisoning. Avoid the use of generic or commercial models for any sensitive application.
- **Do not allow CU**. Currently, agents are nowhere near reliable enough to act as proxies for the human user on the human's own machine.
- **Restrict how much code is written directly by LLMs and agents**. Army's codebase should be transparent to human auditing. That means code written by humans. Coding assistants can be useful, but empowering agents to write code independently is risky.
- **Prohibit agent-to-agent communication**. Currently, NL-prompted agents alone are unreliable. In multi-agent systems, this creates a threat of prompt infection, cascading ambiguity, or other unforeseen consequences that are not yet well understood.
- **Beware of continual learning**. If continual learning becomes possible, Army should refrain from integrating it until matching safety measures have been invented. Even session-based memory (how models 'remember' earlier interactions in a given session) should be restricted until better auditing procedures are in place. For now, freeze models at deployment.

- **Apply existing C2 principles**. Methods to reduce ambiguity are well developed within existing C2 principles used by Army. Concepts such as trust, responsibility and verifiability that already apply to H2H should apply equally to LLM and agent oversight (i.e. to the human operators of these systems). This should include existing accountability structures, procedures for delegated authority, and even procedures for permissions and rank. Ultimately, though, agents themselves cannot be incorporated into trust or rank networks because they are not liable, responsible or punishable.
- **Require human-in-the-loop decision-making for any agent output that may result in external action**. If agents are permitted to control other systems, their actions should be monitored and subject to sign-off by a human. If they produce recommendations or analyses, their proposed actions should not be carried out by other agents and should be overseen by humans.

/ CONCLUSION

The near future will see continued investment in—and adoption of—LLMs and agents. This paper asserts that while the models will improve, the medium (NL) will not. This is a big claim. It runs counter to the hype from Big Tech and the widespread assumption that LLMs and agents will simply increase in general competence from here on. However, *if* it is true that NL involves too much ambiguity and context-dependency ever to work as a command language, *then* it will be impossible to have NL-prompted agents that can reliably be commanded or command others.

The Australian Army already knows the risks of NL for command, having spent decades developing C2 protocols to contain NL's weaknesses. The lesson is that NL can only be partially tamed: miscommunication and the risk of deception are ever-present. Applying this insight to AI, LLMs and agents may offer productivity gains in low-stakes domains but cannot be trusted where stakes are high. They cannot be incorporated into trust and accountability structures in the way humans can, nor can their susceptibility to manipulation ever be totally eradicated.

NL's evolutionary limits need to be acknowledged. Army should therefore exploit the advantages of LLMs and agents while strongly restricting their use. Summarisation, translation and other administrative or research tasks are fertile ground for innovation. But Army must draw a hard line against their use in battlefield decision-making or autonomous operations.

/ ABOUT THE AUTHOR

Dr Jamie Freestone is a research associate at the Australian National University's School of Engineering. His primary field of research is the philosophy of autonomous systems, including agents, humanoid robots, self-driving vehicles, and sleepwalkers. He also works as a consultant on AI safety policy. He was a 2024–25 Australian Army Research Centre Fellow.

/ ACKNOWLEDGMENTS

I would like to thank the team at the Australian Army Research Centre for their help during research and publication. I also thank the anonymous reviewers whose comments greatly improved this paper. And thanks to all the scholars on prompt injection and the evolution of language whom I reached out to, especially Jean-Louis Dessalles, Xiaogeng Liu, Steven Pinker, Thom Scott-Phillips, and Yotam Wolf.

/ ENDNOTES

1 Emelia Probasco, Matthew Burtell, Helen Toner and Tim GJ Rudner, 'Not Oracles of the Battlefield: Safety Considerations for AI-Based Military Decision Support Systems', *Proceedings of the AAAI/ ACM Conference on AI, Ethics, and Society* 7, no. 1 (2024): 1157–1165, at: https://dl.acm.org/doi/10.5555/3716662.3716763.

2 Evan Lynch, 'U.S. Army Gives Update on Generative Artificial Intelligence and Large Language Models', *Signal*, 20 August 2024, at: https://www.afcea.org/signal-media/us-army-gives-update-generative-artificial-intelligence-and-large-language-models.

3 Firms that are offering products to the military include Anduril, Booz Allen, Leidos, Palantir, Scale AI and Shield AI. See also Sam Biddle, 'U.S. Military Makes First Confirmed OpenAI Purchase for War-Fighting Forces', *The Intercept*, 25 October 2024, at: https://theintercept.com/2024/10/25/africom-microsoft-openai-military.

4 Alex Karp, 'Bending Artificial Intelligence to Our Collective Will: A Letter from the Chief Executive Officer', *Palantir*, 7 April 2023, at: https://www.palantir.com/newsroom/letters/our-new-platform; Palantir AIP, 'Defense and Military' (video), *YouTube*, 26 April 2023, at: https://www.youtube.com/watch?v=XEM5qz__HOU.

5 Timothy Papandreou, '2025: Agentic and Physical AI—a Multitrillion Dollar Industry', *Forbes*, 15 January 2025, at: https://www.forbes.com/sites/timothypapandreou/2025/01/15/2025-agentic--physical-aia-multi-trillion-dollar-economy-emerges. Papandreou is referring to Nvidia CEO Jenson Huang's public statement. The word *agent* has many meanings in many disciplines. These are the early days of such technology and the terminology may shift. At the time of writing, agents are sometimes known as *large model* (*LM*) *agents* or *agentic AI*, but most commonly just *agents* or *AI agents*.

6 For a scholarly survey, see Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun and Yue Zhang, 'A Survey on Large Language Model (LLM) Security and Privacy: The Good, the Bad, and the Ugly', *High-Confidence Computing* 4, no. 2 (2024): 100211, at: https://doi.org/10.1016/j.hcc.2024.100211. For an incisive critique of the use of these systems in a military decision-making context, see Cameron Hunter and Bleddyn E Bowen, 'We'll Never Have a Model of an AI Major-General: Artificial Intelligence, Command Decisions, and Kitsch Visions of War', *Journal of Strategic Studies* 47, no. 1 (2024): 116–146, at: https://doi.org/10.1080/01402390.2023.2241648.

7 Yao et al., 'A Survey on Large Language Model (LLM) Security and Privacy'.

8 Subbarao Kambhampati et al., 'Position: LLMs Can't Plan, but Can Help Planning in LLM-Modulo Frameworks', Proceedings of the 41st International Conference on Machine Learning, in *Proceedings of Machine Learning Research* 235 (2024): 22895–22907, at: https://proceedings.mlr.press/v235/kambhampati24a.html; Sean Williams and James Huckle, 'Easy Problems That LLMs Get Wrong', *arXiv* (2024), at: https://arxiv.org/abs/2405.19616; Tarun Gupta et al., 'The Essential Role of Causality in Foundation World Models for Embodied AI', *arXiv* (2024), at: https://arxiv.org/abs/2402.06665.

9 Anat Lior, 'AI Entities as AI Agents: Artificial Intelligence Liability and the AI Respondeat Superior Analogy', *Mitchell Hamline Law Review* 46, no. 5 (2019): 1043–1102, at: https://open.mitchellhamline.edu/mhlr/vol46/iss5/2.

10 For intelligence analysis, see Sarah Logan, 'Tell Me What You Don't Know: Large Language Models and the Pathologies of Intelligence Analysis', *Australian Journal of International Affairs* 78, no. 2 (2024): 220–228, at: https://doi.org/10.1080/10357718.2024.2331733. For resort-to-war decisions, see Toni Erskine, '*Before* Algorithmic Armageddon: The Erosion of Restraint as an Immediate Risk of AI Infiltrating the Decision to Wage War', *Australian Journal of International Affairs* 78, no. 2 (2024): 175–190, at: https://doi.org/10.1080/10357718.2024.2345636.

11 Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz and Mario Fritz, 'Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection', *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security* (2023), pp. 79–90, at: https://doi.org/10.1145/3605764.3623985.

12 The technical literature on the nature of LLMs and agents, as well as their security risks, is very recent, most of it having been written since 2023. Much of it hasn't even been peer reviewed yet. Of necessity, many works cited here are preprints taken from arXiv, the main repository for such work.

13 The canonical work in this line of research is Richard Dawkins and John R Krebs, 'Arms Races Between and Within Species', *Proceedings of the Royal Society of London, Series B Biological Sciences* 205, no. 1161 (1979): 489–511, at: https://doi.org/10.1098/rspb.1979.0081. See also Donald H Owings and Eugene S Morton, *Animal Vocal Communication: A New Approach* (Cambridge University Press, 1998).

14 Mary J West-Eberhard, 'Sexual Selection, Competitive Communication and Species-Specific Signals in Insects', in Trevor Lewis (ed.), *Insect Communication*: *Proceedings of the 12th Symposium of the Royal Entomological Society of London* (Academic Press, 1984), pp. 283–324, at: https://repository.si.edu/server/api/core/bitstreams/49964b5c-d117-474d-b1a0-8fe62bc43425/content.

15 Marc D Hauser, *The Evolution of Communication* (MIT Press, 1996), Section 2.2.

16 Jean-Louis Dessalles, *Why We Talk: The Evolutionary Origins of Language* (Oxford University Press, 2007).

17 James R Hurford, *The Origins of Grammar: Language in the Light of Evolution* (Oxford University Press, 2012), pp. 96–97.

18 Theo MV Janssen and Barbara H Partee, 'Compositionality', in Johan van Benthem and Alice ter Meulen (eds), *Handbook of Logic and Language* (North-Holland, 1997), pp. 417–473. For compositionality in the language of humans *and* LLMs, see Jacob Russin, Sam W McGrath, Danielle J Williams and Lotem Elber-Dorozko, 'From Frege to chatGPT: Compositionality in Language, Cognition, and Deep Neural Networks', *arXiv* (2024), at: https://arxiv.org/abs/2405.15164.

19 Dessalles, *Why We Talk*, p. 18.

20 Thom Scott-Phillips, *Speaking Our Minds: Why Human Communication Is Different, and How Language Evolved to Make It Special* (Bloomsbury Publishing, 2014), pp. 1–3.

21 There are many other ways that context, broadly conceived, aids in supplementing the meaning of the words in a message. Many of these are components of how the message is delivered, or the *pragmatics* of language, including prosody, body language, gaze etc. See Stephen C Levinson, *The Dark Matter of Pragmatics: Known Unknowns* (Cambridge University Press, 2024).

22 Dessalles, *Why We Talk*.

23 Terrence W Deacon, *The Symbolic Species: The Co-Evolution of Language and the Brain* (WW Norton & Company, 1997), p. 52; Robin Dunbar, *Grooming, Gossip, and the Evolution of Language* (Harvard University Press, 1996).

24 Daniel Dor, *The Instruction of Imagination: Language as a Social Communication Technology* (Oxford University Press, 2015), pp. 210–212; Nathan Oesch, 'Deception as a Derived Function of Language', *Frontiers in Psychology* 7 (2016): 1485, at: https://doi.org/10.3389/fpsyg.2016.01485; Thom Scott-Phillips, 'Why Talk? Speaking as Selfish Behaviour', in Angelo Cangelosi, Andrew DM Smith and Kenny Smith (eds), *The Evolution of Language*: *Proceedings of the 6th International Conference (EVOLANG6)* (World Scientific, 2006), pp. 299–306.

25 James Johnson, 'Automating the OODA Loop in the Age of Intelligent Machines: Reaffirming the Role of Humans in Command-and-Control Decision-Making in the Digital Age', *Defence Studies* 23, no. 1 (2023): 43–67, at: https://doi.org/10.1080/14702436.2022.2102486.

26 'P10-37 Basics: Intro & Brevity', *United Task Force* (website), 2025, at: https://unitedtaskforce.net/training/sop/communication/basics-brevity.

27 Ulrich Schade, Bastian Haarmann and Michael R Hieb, 'A Grammar for Battle Management Language', *2011 IEEE/ACM 15th International Symposium on Distributed Simulation and Real Time Applications* (2011), pp. 155–159, at: https://www.computer.org/csdl/proceedings/ds-rt/2011/12OmNC3Xhik.

28 See for example 'Formal and Natural Languages', in *How to Think Like a Computer Scientist: Interactive Edition* (Runestone Academy), at: https://runestone.academy/ns/books/published/thinkcspy/GeneralIntro/FormalandNaturalLanguages.html (accessed 20 November 2024).

29 Paula Buttery, *Formal Models of Language: Formal Versus Natural Language* (University of Cambridge Computer Science, 2020), at: https://www.cl.cam.ac.uk/teaching/2021/ForModLang/notes/Formal_vs_Natural.pdf; Kevin DeLaplante, 'The Difference Between Natural Languages and Formal Languages', *Critical Thinker Academy* (website), at: https://criticalthinkeracademy.teachable.com/courses/2514/lectures/761246.

30 Steven Pinker, *The Language Instinct: How the Mind Creates Language* (Camberwell: Penguin Australia, 2003), pp. 78–81.

31 Because ambiguity is generally resolved by context, we could further compress this by saying that all these features of NL relate to context dependency. Although this is probably sound, ambiguity is worth retaining as a separate feature because it is widely identified by different commentators from different disciplines and its mitigation strategies are also usefully separated from those of context dependency. See Section 5.

32 Edsger W Dijkstra, 'Some Comments on the Aims of MIRFAC', *Communications of the ACM* 7, no. 3 (1964): 190, at: https://dl.acm.org/doi/pdf/10.1145/363958.364002; Edsger W Dijkstra, 'On the Foolishness of "Natural Language Programming"', in Friedrich Bauer et al. (eds), *Program Construction: International Summer School* (Springer Berlin, 1979), pp. 51–53; ID Hill, 'Wouldn't It Be Nice If We Could Write Computer Programs in Ordinary English—Or Would It?', *Honeywell Computer Journal* 6, no. 2 (1972): 76–83, at: https://doi.org/10.1093/combul/15.6.306; Lance A Miller, 'Natural Language Programming: Styles, Strategies, and Contrasts', *IBM Systems Journal* 20, no. 2 (1981): 184–215, at: https://doi.org/10.1147/sj.202.0184.

33 Pioneers of NL programming also worried about punctuation and the representation of numbers, although LLMs handle these with aplomb; see Miller, 'Natural Language Programming'. And contemporary researchers raise other NL-related problems not entailed by the framework offered here, such as the tangled hierarchies present in NL but not PL; see Advait Sarkar, Andrew D Gordon, Carina Negreanu, Christian Poelitz, Sruti S Ragavan and Ben Zorn, 'What Is It Like to Program with Artificial Intelligence?', *arXiv* (2022), at: https://arxiv.org/abs/2208.06213.

34 John F Pane and Brad A Myers, 'Studying the Language and Structure in Non-Programmers' Solutions to Programming Problems', *International Journal of Human-Computer Studies* 54, no. 2 (2001): 237–264, at: https://doi.org/10.1006/ijhc.2000.0410.

35 Johnson, 'Automating the OODA Loop in the Age of Intelligent Machines'.

36 For a rare study that goes beyond the dyadic sender–receiver model, see L'udovít Malinovský, 'Broadcasting to the Enemy: Deception as a Solution in Evolution of Language', in Erica A Cartmill, Seán Roberts, Heidi Lyn and Hannah Cornish (eds), *Evolution of Language: Proceedings of the 10th International Conference (EVOLANG10)* (World Scientific, 2014), pp. 169–176.

37 Giorgia Mannaioli, Alessandro Ansani, Claudia Coppola and Edoardo L Vallauri, 'Vagueness as an Implicit-Encoding Persuasive Strategy: An Experimental Approach', *Cognitive Processing* 25, no. 2 (2024): 205–227; Steven Pinker, *The Stuff of Thought: Language as a Window Into Human Nature* (Penguin, 2007), pp. 423–425.

38 There is perhaps an analogy here to *declarative* PLs—as opposed to standard PLs, which are by contrast *imperative* PLs. Declarative PLs allow a user to, for example, query a database by specifying *what* results they want, rather than having to write the imperative code that would actually instruct the machine *how* to find what they want. This allows ordinary users to learn a relatively simple PL with a closer resemblance to NL. See Edgar F Codd, 'A Relational Model of Data for Large Shared Data Banks', *Communications of the ACM* 13, no. 6 (1970): 377–387, at: https://doi.org/10.1145/362384.362685.

39 Desirae Gieseman, 'Effective Writing for Army Leaders: The Army Writing Standard Redefined', *Military Review* 95, no. 5 (2015): 106–118, at: https://www.armyupress.army.mil/Portals/7/military-review/Archives/English/MilitaryReview_20151031_art016.pdf.

40 Maarten Schadd, Anne M Sternheim, Romy Blankendaal, Martin van der Kaaij and Olaf Visker, 'How a Machine Can Understand the Command Intent', *The Journal of Defense Modeling and Simulation* 22, no. 1 (2025): 41–58, at: https://doi.org/10.1177/15485129221115736.

41 Ibid., p. 42.

42 Schade et al., 'A Grammar for Battle Management Language'.

43 Ibid.

44 Alexander Wei, Nika Haghtalab and Jacob Steinhardt, 'Jailbroken: How Does LLM Safety Training Fail?', *Advances in Neural Information Processing Systems* 36 (2024): 80079–80110, at: https://doi.org/10.52202/075280-3508.

45 Sara Abdali, Richard Anarfi, CJ Barberan and Jia He, 'Securing Large Language Models: Threats, Vulnerabilities and Responsible Practices', *arXiv* (2024), at: https://arxiv.org/abs/2403.12503; Yi Zeng, Hongpeng Lin, Jingwen Zhang, Diyi Yang, Ruoxi Jia and Weiyan Shi, 'How Johnny Can Persuade LLMs to Jailbreak Them: Rethinking Persuasion to Challenge AI Safety by Humanizing LLMs', *arXiv* (2024), at: https://arxiv.org/abs/2401.06373.

46 Fábio Perez and Ian Ribeiro, 'Ignore Previous Prompt: Attack Techniques for Language Models', *arXiv* (2022), at: *https://arxiv.org/abs/2211.09527*.

47 Maximilian AJ Mozes, *Understanding and Guarding Against Natural Language Adversarial Examples*, PhD thesis, University College London, 2024, at: https://discovery.ucl.ac.uk/id/eprint/10190224; Sippo Rossi, Alisia M Michel, Raghava R Mukkamala and Jason B Thatcher, 'An Early Categorization of Prompt Injection Attacks on Large Language Models', *arXiv* (2024), at: https://arxiv.org/abs/2402.00898.

48 Greshake et al., 'Not What You've Signed Up For'. See also U Anwar et al., 'Foundational Challenges in Assuring Alignment and Safety of Large Language Models', *arXiv* (2024), Section 3.5.7, at: https://arxiv.org/abs/2404.09932.

49 Another way of putting the same problem: 'the core issue is that, contrary to standard security best practices, "control" and "data" planes are *not separable* when working with LLMs. A single prompt contains both control and data. The prompt injection technique exploits this lack of separation to insert control elements where data is expected, and thus enables attackers to reliably control LLM outputs'. See Rich Harang, 'Securing LLM Systems Against Prompt Injection', *Nvidia Developer* (website), 3 August 2023, at: https://developer.nvidia.com/blog/securing-llm-systems-against-prompt-injection. Note that there is a similarity here to other attacks that exploit a system with a common channel for data and instructions. SQL injection attacks do so with database queries, and many techniques in phreaking (hacking phone networks) worked according to the same principle. However, PI represents an especially vulnerable blurring of data and control because the medium is NL, which is open-ended, usable by amateurs and highly ambiguous. I thank an anonymous reviewer for this salutary comparison.

50 Greshake et al., 'Not What You've Signed Up For'.

51 Ibid.

52 Nicholas Carlini et al., 'Poisoning Web-Scale Training Datasets Is Practical', *2024 IEEE Symposium on Security and Privacy (SP)* (2023), pp. 407–425, at: https://doi.org/10.1109/SP54263.2024.00179.

53 Xiangyu Zhou et al., 'Learning to Poison Large Language Models During Instruction Tuning', *arXiv* (2024), at: https://arxiv.org/abs/2402.13459.

54 At the time of writing, there was no Wikipedia article dedicated to agents, despite the beta release of OpenAI's Operator and Anthropic's CU function for its LLM Claude.

55 Papandreou, '2025: Agentic and Physical AI'.

56 See Mitra Sorrells, 'OpenAI Debuts 'Operator' Agent That Can Book Travel', *PhocusWire*, 24 January 2025, at: https://www.phocuswire.com/openai-operator-web-agent; Antonio Pequeño, 'From Grocery Orders to Ticket Purchases, OpenAI Unveils Its $200-a-Month "Operator"', *Forbes*, 24 January 2025, at: https://www.forbes.com.au/news/innovation/openaioperator-can-now-order-groceries-make-reservations; Will D Heaven, 'OpenAI Launches Operator—an Agent That Can Use a Computer for You', *MIT Technology Review,* 23 January 2025, at: https://www.technologyreview.com/2025/01/23/1110484/openai-launches-operator-an-agent-that-can-use-a-computer-for-you.

57 Even this is proving difficult. See Peter Cohan, 'Salesforce Stock Down as AI Agents Aim to Win Skeptical Customers', *Forbes*, 14 February 2025, at: https://www.forbes.com/sites/petercohan/2025/02/14/salesforce-stock-falls-as-ai-agents-aim-to-win-skeptical-customers.

58 See Vittoria Dentella, Fritz Günther, Elliot Murphy, Gary Marcus and Evelina Leivada, 'Testing AI on Language Comprehension Tasks Reveals Insensitivity to Underlying Meaning', *Scientific Reports* 14, no. 1 (2024): 28083, at: https://doi.org/10.1038/s41598-024-79531-8; Melissa Heikkila and Will D Haven, 'Anthropic's Chief Scientist on 4 Ways Agents Will Be Even Better in 2025', *MIT Technology Review*, 11 January 2025, at: https://www.technologyreview.com/2025/01/11/1109909/anthropics-chief-scientist-on-5-ways-agents-will-be-even-better-in-2025; Helen Toner et al., *Through the Chat Window and Into the Real World: Preparing for AI Agents* (Center for Security and Emerging Technology, 2024), at: https://cset.georgetown.edu/publication/through-the-chat-window-and-into-the-real-world-preparing-for-ai-agents.

59 Fangzhou Wu, Ning Zhang, Somesh Jha, Patrick McDaniel and Chaowei Xiao, 'A New Era in LLM Security: Exploring Security Concerns in Real-World LLM-Based Systems', *arXiv* (2024), at: https://arxiv.org/abs/2402.18649.

60 Donghyun Lee and Mo Tiwari, 'Prompt Infection: LLM-to-LLM Prompt Injection Within Multi-Agent Systems', *arXiv* (2024), at: https://arxiv.org/abs/2410.07283.

61 Wu et al., 'A New Era in LLM Security'; Yuhao Wu, Franziska Roesner, Tadayoshi Kohno, Ning Zhang and Umar Iqbal, 'IsolateGPT: An Execution Isolation Architecture for LLM-Based Agentic Systems', Network and Distributed System Security (NDSS) Symposium 2025, 24–28 February 2025, *NDSS* (website), at: https://www.ndss-symposium.org/ndss-paper/isolategpt-an-execution-isolation-architecture-for-llm-based-agentic-systems.

62 Also known as a computer controlling agent (CCA) or a graphical user interface (GUI) agent.

63 There is not much high-quality research on this yet, but early indications are that coding assistants are highly productive, at least for routine coding tasks. See Thomas Weber, Maximilian Brandmaier, Albrecht Schmidt and Sven Mayer, 'Significant Productivity Gains Through Programming with Large Language Models', *Proceedings of the ACM on Human-Computer Interaction* 8, no. EICS (2024): 1–29, at: https://dl.acm.org/doi/abs/10.1145/3661145.

64 Vrunda Gadesha, 'What Is One Shot Prompting?', *IBM* (website), at: https://www.ibm.com/think/topics/one-shot-prompting (accessed 14 February 2025).

65 Siyuan Hu, Mingyu Ouyang, Difei Gao and Mike Zheng Shou, 'The Dawn of GUI Agent: A Preliminary Case Study with Claude 3.5 Computer Use, *arXiv* (2024), at: https://arxiv.org/abs/2411.10323; Shuai Wang et al., 'GUI Agents with Foundation Models: A Comprehensive Survey', *arXiv* (2024), at: https://arxiv.org/abs/2411.04890.

66 Companies like Slack and ServiceNow already facilitate hybrid human–agent teams. This is a technology in its infancy, but is envisioned to grow markedly (although it may be hampered by the problems outlined in this article). See *2025: The Year the Frontier Firm Is Born* (Microsoft, 2025), at: https://www.microsoft.com/en-us/worklab/work-trend-index/2025-the-year-the-frontier-firm-is-born.

67 Greshake et al., 'Not What You've Signed Up For'; Y Liu et al., 'Prompt Injection Attack Against LLM-Integrated Applications', *arXiv* (2023), at: https://arxiv.org/abs/2306.05499.

68 Fanlong Zeng, Wensheng Gan, Yongheng Wang, Ning Liu and Philip S Yu, 'Large Language Models for Robotics: A Survey', *arXiv* (2023), at: https://arxiv.org/abs/2311.07226.

69 Alexander Robey, Zachary Ravichandran, Vijay Kumar, Hamed Hassani and George J Pappas, 'Jailbreaking LLM-Controlled Robots', *arXiv* (2023), at: https://arxiv.org/abs/2410.13691.

70 There is much research on how easily persuaded LLMs are, but not in direct comparison to humans. It is a prediction of this article that humans are less persuadable than current LLMs. For the many psychological defences humans have against being persuaded or brainwashed, see Hugo Mercier, *Not Born Yesterday: The Science of Who We Trust and What We Believe* (Princeton University Press, 2020).

71 Jean-Louis Dessalles (personal communication); Thom Scott-Phillips (personal communication); Steven Pinker (personal communication).

72 Xiaogeng Liu (personal communication); Yotam Wolf (personal communication). See also Yotam Wolf, Noam Wies, Oshri Avnery, Yoav Levine and Amnon Shashua, 'Fundamental Limitations of Alignment in Large Language Models', *arXiv* (2023), at: https://arxiv.org/abs/2304.11082. For a survey of experts' predictions of the 'jailbreakability' of future systems, see Katja Grace, Harlan Stewart, Julia F Sandkühler, Stephen Thomas, Ben Weinstein-Raun and Jan Brauner, 'Thousands of AI Authors on the Future of AI', *arXiv* (2024), p. 11, at: https://arxiv.org/abs/2401.02843.

73 Competitions are frequently held to help developers detect bugs in systems. The jailbreaking of LLMs, however, is more of a *fait accompli*. See for example 'Break AI. Win Prizes. Get Discovered', *Grey Swan Arena* (website), at: https://app.grayswan.ai/arena. See also Microsoft's prize-offering 'LLMail-Inject: Adaptive Prompt Injection Challenge', *Microsoft Open Source* (website), at: https://microsoft.github.io/llmail-inject.

74 There are many guides being produced for 'prompt engineering' techniques for LLMs. See for example 'Advanced Prompt Engineering with Structured JSON: Optimizing LLM Interactions', *Modular Resources*, at: https://www.modular.com/ai-resources/advanced-prompt-engineering-with-structured-json-optimizing-llm-interactions?utm_source=chatgpt.com.

75 Yotam Wolf, Noam Wies, Dorin Shteyman, Binyamin Rothberg, Yoav Levine and Amnon Shashua, 'Tradeoffs Between Alignment and Helpfulness in Language Models', *arXiv* (2024), at: https://arxiv.org/abs/2401.16332. And see a recent example of universal prompt injection technique that uses a prompt in an XML format to fool the model into accepting it as a policy or instruction prompt rather than a normal user query: Conor McCauley, Kenneth Yeung, Jason Martin and Kasimir Schulz, 'Novel Universal Bypass for All Major LLMs', *HiddenLayer*, 24 April 2025, at: https://hiddenlayer.com/innovation-hub/novel-universal-bypass-for-all-major-llms (accessed 10 July 2025).

76 Schadd et al., 'How a Machine Can Understand the Command Intent'.

77 Fangwen Mu et al., 'ClarifyGPT: Empowering LLM-Base Code Generation with Intention Clarification', *arXiv* (2023), at: https://arxiv.org/abs/2310.10996.

78 Davinder Kaur, Suleyman Uslu, Kaley J Rittichier and Arjan Durresi, 'Trustworthy Artificial Intelligence: A Review', *ACM Computing Surveys* (*CSUR*) 55, no. 2 (2022): 1–38, at: https://dl.acm.org/doi/10.1145/3491209; Gert-Jan Lokhorst and Jeroen van den Hoven, 'Responsibility for Military Robots', in Patrick Lin, Keith Abney and George A Bekey (eds), *Robot Ethics: The Ethical and Social Implications of Robotics* (The MIT Press, 2012), pp. 145–156.

79 Robert Sparrow, 'Killer Robots', *Journal of Applied Philosophy* 24, no. 1 (2007): 62–77, at: https://robsparrow.com/wp-content/uploads/Killer-robots.pdf; Kai C Yam et al, 'Robots at Work: People Prefer—and Forgive—Service Robots with Perceived Feelings', *Journal of Applied Psychology* 106, no. 10 (2021): 1557–1572, at: https://doi.org/10.1037/apl0000834.

80 Wolf et al., 'Tradeoffs Between Alignment and Helpfulness in Language Models'; Zeng et al., 'How Johnny Can Persuade LLMs to Jailbreak Them'.

81 GovTech Data Science and Singapore Government AI Division, *Prompt Engineering Playbook* (2023), p. 26, at: https://www.developer.tech.gov.sg/products/collections/data-science-and-artificial-intelligence/playbooks/prompt-engineering-playbook-beta-v3.pdf.

82 At the time of writing, there is no legislation governing autonomous decision-making processes, but there are widespread calls for guidelines and laws for military and civilian uses. See for example Australian Government et al., *National Framework for the Assurance of Artificial Intelligence in Government* (Commonwealth of Australia, 2024), at: https://www.finance.gov.au/sites/default/files/2024-06/National-framework-for-the-assurance-of-AI-in-government.pdf. See also Commonwealth Ombudsman, *Better Practice Guide: Automated Decision Making* (March 2025), at: https://www.ombudsman.gov.au/__data/assets/pdf_file/0025/317437/Automated-Decision-Making-Better-Practice-Guide-March-2025.pdf.

83 Naveen F Awad and Mayuram S Krishnan, 'The Personalization Privacy Paradox: An Empirical Evaluation of Information Transparency and the Willingness to Be Profiled Online for Personalization', *MIS Quarterly* 30, no. 1 (2006): 13–28, at: https://doi.org/10.2307/25148715.

84 Bianca Kronemann, Hatice Kizgin, Nripendra Rana and Yogesh K Dwivedi, 'How AI Encourages Consumers to Share Their Secrets? The Role of Anthropomorphism, Personalisation, and Privacy Concerns and Avenues for Future Research', *Spanish Journal of Marketing—ESIC* 27, no. 1 (2023): 3–19, at: https://doi.org/10.1108/SJME-10-2022-0213.

85 Sewon Min et al., 'Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?', *arXiv* (2022), at: https://arxiv.org/abs/2202.12837.

86 Sebastian Thrun, 'Lifelong Learning Algorithms', in Sebastian Thrun and Lorien Pratt (eds), *Learning to Learn* (Boston: Springer US, 1998), pp. 181–209; Liyuan Wang, Xingxing Zhang, Hang Su and Jun Zhu, 'A Comprehensive Survey of Continual Learning: Theory, Method and Application', *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46, no. 8 (2024): 5362–5383, at: https://doi.org/10.1109/TPAMI.2024.3367329.

87 Rob Toews, 'The Gaping Hole in Today's AI Capabilities', *Forbes*, 23 March 2025, at: https://www.forbes.com/sites/robtoews/2025/03/23/the-gaping-hole-in-todays-ai-capabilities-1.

88 Rishi Bommasani et al., 'On the Opportunities and Risks of Foundation Models', *arXiv* (2021), at: https://arxiv.org/abs/2108.07258.

89 This may be especially true for AI-based systems. See Robert Williams and Roman Yampolskiy, 'Understanding and Avoiding AI Failures: A Practical Guide', *Philosophies* 6, no. 3 (2021): 53, at: https://doi.org/10.3390/philosophies6030053.

90 Svetoslav Gaidow and Seng Boey, *Australian Defence Risk Management Framework: A Comparative Study* (Commonwealth of Australia, 2005), at: https://apps.dtic.mil/sti/pdfs/ADA434592.pdf.

91 Australian Army, *Robotic & Autonomous Systems Strategy v2.0*, 11 August 2022, at: https://researchcentre.army.gov.au/rico/robotic-and-autonomous-systems/robotic-autonomous-systems-ras-strategy.

92 Commonwealth Ombudsman, 'Better Practice Guide'.

RESEARCHCENTRE.ARMY.GOV.AU

CG: JUN002-26