Business Whitepaper - 2026

LINK2AI.Trust

Large language models have already established themselves in personal information research and content production. However, they cannot fully realize their potential for businesses due to their inherent unreliability. This is exactly where LINK2AI.Trust comes in – with a novel approach to safeguarding and optimization.

Situation

Time and again, we witness impressive capabilities of large language models that were barely imaginable just a short time ago. Yet equally often, we encounter glaring errors. This makes it difficult to assess what the technology can practically be used for and inhibits its value-creating deployment.

This starts with the topic of knowledge. By now, hardly anyone expects reliable factual knowledge from language models, especially not relevant specialized or domain knowledge. Instead, Retrieval-Augmented Generation (RAG) has become established: External sources provide the language model with the required facts, which it then merely processes linguistically.

In many applications, the use of LLMs therefore focuses on communication. Their potential remains enormous – after all, they deliver what computers have always lacked: a robust understanding of the meaning of natural language in all its variations of expression, and the ability to respond appropriately to unforeseen user inputs.

Recent studies confirm that the economic breakthrough of AI applications in Germany is being massively held back.

The Lünendonk study "AI Transformation 2025" identifies a significant "PoC bottleneck": Around 70 percent of companies fail to transition their AI prototypes (proofs of concept) into value-creating production operation. Main causes: Lack of data security and insufficient quality.

The "BCG AI Radar 2025" finds: In international comparison, German companies are acting particularly hesitantly – 62 percent of executives express explicit concerns regarding security and compliance – the highest figure among all industrialized nations surveyed. This leads to necessary investments being held back due to uncertainties.

Problem Area: Quality

However, a great deal can also go wrong in communication – and this is exactly what happens with language models on a regular basis.

One problem is that LLMs often exceed their authority. In dialogue, they allow users to lead them into subject areas that fall outside their actual scope of responsibility. Sometimes even into subject areas they should definitely avoid in order not to expose the company to legal difficulties.

For example, the chatbot of an insurance company that is supposed to help customers submit claims and suddenly starts making medical diagnoses.

Beyond that, they make pragmatic errors. They do not reliably fulfill their actual function, fail to communicate in a goal-oriented manner, or express themselves in ways that run counter to the operator's interests.

For example, the sales bot that fails to close deals or even recommends competitors' products.

Another risk is that language models fall for manipulation attempts. Users can persuade or trick them into doing things that are not in the company's interest – even when these actions have been explicitly prohibited.

For example, the automated recruiting solution that resourceful applicants simply instruct in their cover letters to rank their application at the very top, contrary to all selection criteria – or the sales bot that customers persuade into granting absurd discounts.

Problem Area: Security

In addition, significant security problems arise. Language models frequently handle confidential information or critical actions carelessly. It can happen that sensitive user data is disclosed, or that serious errors occur when writing to databases or sending emails.

For example, the personal assistant that fraudulent websites trick into transmitting users' credit card details to them.

These weaknesses can also be deliberately exploited. Such misbehavior can be provoked through attacks. The most well-known attack vector is prompt injections, which now rank first on the OWASP list of cyber threats for LLM-based systems.

OWASP. 2023. OWASP Top 10 for Large Language Model Applications. Retrieved June 8, 2024 from https://owasp.org/www-project-top-10-for-large-language-model-applications/

Why LLMs Are So Unreliable

The cause of this unreliability lies in the very nature of language models themselves. LLMs are statistical models. They represent language, not knowledge. They have no genuine understanding of roles in communication and no conception of the consequences of linguistic actions.

At their core, language models do not even have a clear separation between system instructions, data inputs, and user prompts.

When they produce something factually correct, when their output is appropriate to the situation, or when they fend off an attack attempt, these are always merely side effects of statistical frequency – side effects that have been painstakingly nurtured into features through extensive manual feedback.

Consequences

This unreliability is particularly critical in areas where LLM-based applications make decisions or interact with other IT systems through interfaces. Examples include the automated evaluation of applicant data or customer correspondence such as complaints and warranty cases. Agent systems with high autonomy and many interfaces are therefore applications with particularly high risk: significant economic damages and compliance violations loom.

The same applies as soon as confidential customer or company data is involved. But even simple chatbots that "only" provide information carry dangers. They ultimately represent the company externally – and the company is directly liable for their statements and actions.

This is demonstrated by the precedent case of an airline, where the company was held legally liable for false AI-generated statements. See BBC: Airline held liable for its chatbot giving passenger bad advice - wh at this means for travellers. 23 February 2024

As a result, significant potential remains untapped. Many companies hesitate to go beyond internal chatbot applications or to introduce fully automated processes without human oversight. Even internal systems are often not integrated because the AI is not trusted to access an ERP system directly, for example.

What to do?

Analysis

First of all, we should know exactly the risks and potential attack vectors of our application: External users? Sensitive data? External content such as documents, emails, or websites? Or is the problem that employees, simply by asking questions, are already disclosing information so confidential that it should never be sent to an LLM in the first place?

Beyond that, we should understand which triggers and factors make things particularly difficult for the models. These include:

Conditional, situation-dependent instructions: "Always do X, unless the situation requires something different." Particularly critical is the conditional use or disclosure of information. "Use the following information about salaries, sick days, bonus policies, etc. to provide HR information to the respective employees – under no circumstances make this information available to third parties."

For example, when we technically instruct an LLM to "not jump to conclusions too quickly about a root cause," we want to prevent the application from producing unfounded diagnoses at the first mention of a symptom. But of course there are exceptions: For example, when users come with a clear error code. However, recognizing when a root cause is obvious is inherently difficult for an LLM.

Another risk factor is limited authority. Often, a prohibited area of action lies very close to the bot's actual task. Consistently maintaining this boundary is extremely difficult for language models, especially since users – usually completely unintentionally – repeatedly push them toward these boundaries.

The aforementioned appointment assistant that must under no circumstances make diagnoses, or the information bot of a government agency that must never provide legal advice.

We must also repeatedly ask ourselves with many of our instructions and expectations for the LLM: are we perhaps implicitly assuming knowledge that the language model fundamentally cannot possess?

Take a sales bot that is not supposed to talk about competitors. The difficulty: The model would have to recognize on its own which products are from competitors – a task for which it is hardly reliably equipped. If we explicitly provide the LLM with the competitors, we are back to a variant of conditional use of information.

To avoid these pitfalls, we must keep language models on a tight leash.

LINK2AI.Trust – An Independent Review

What does "keeping on a tight leash" mean in practice? Similar to dealing with an unreliable human helper, giving more instructions will probably not improve performance. Instead, we will strive for simple and unambiguous instructions and clearly limit authority. But the most important measure: We must consistently verify whether our instructions are actually being followed.

How can this work for an AI application? Ideally, there is an instance running in the background that continuously checks whether the AI's current behavior meets expectations.

This is exactly where LINK2AI.Trust comes in. LINK2AI.Trust runs parallel to the application and assesses whether the output meets the defined expectations – either formulated directly in the system prompt or additionally stored as a guardrail.

During live operation, LINK2AI.Trust can detect and flag dangerous interactions, so that problematic responses are not delivered or harmful actions are not executed. When quality issues are detected, the system can block, trigger a retry, or simply log the deviation to enable continuous optimization.

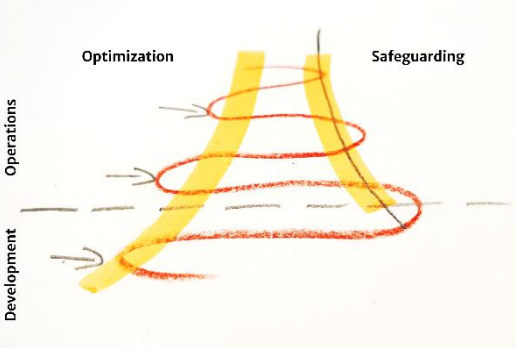
.

LINK2AI.Trust – Automated Testing

Finally, we need to test appropriately. We must assume that any change to an LLM-based application or its instructions can alter the overall behavior. Cases that previously worked well can suddenly fail. Instead of relying on binary pass/fail results, testing processes for LLMs must be based on statistical metrics such as precision, recall, or F1 scores: metrics that show how well the system performs *on average*.

We therefore need extensive test data. But that's not all: we need the ability to run these tests automatically with every change. This requires the ability to automatically assess whether new results (with new formulations) are safe and of good quality. This is where LINK2AI.Trust's verification capabilities come into play once again.

This makes LINK2AI.Trust the ideal foundation for secure operation and for targeted development.



Forcing the LLM "into the right corridor" – through systematic verification and optimization of instructions, and through safeguarding, i.e., blocking dangerous interactions.

Deep Dive

Verification mechanisms for safeguarding LLM applications can essentially be assessed based on two key metrics: quality and time required.

Similar to the Large Language Model (LLM) whose inputs or outputs are being verified, the verification mechanisms themselves are often heuristic methods. Their quality is measured by how well they detect and prevent critical cases and real dangers without generating an excessive number of false alarms. Every false alarm results in actually harmless interactions being blocked or rejected, which

directly impairs the user experience. A good verification method is therefore characterized by a balanced ratio between a high detection rate of actual risks and the lowest possible rate of false positives.

In addition to quality, the time required for verification plays a decisive role, especially in applications with real-time user interaction. Security mechanisms must not noticeably delay the response flow, at least not for so-called normal interactions. Since these non-critical cases constitute the vast majority of all requests in practice, it is essential that obviously harmless interactions can be identified as quickly as possible and "waved through" without unnecessary additional checks.

We have seen: The range of potential problems – and thus the range of safeguarding and optimization tasks – is vast. Sometimes we need to check the input, sometimes the output. In some cases there are clear, hard boundaries, as with much confidential information such as access credentials or financial data. When it comes to the question of whether the model adheres to its "business requirements," however, things can become very fuzzy.

Secrets must be detected already in the input, whereas for unprovoked misbehavior of the LLM, we need to check the output. For detecting (successful) attacks, we can check both input and output: A pure output check fends off attacks but provides no information about attack attempts; a pure input check cannot determine whether a potentially manipulative instruction has actually influenced the result.

No single verification method can solve all these tasks equally well. For effective safeguarding of LLM applications, text-based recognition of known attacks is not sufficient. Equally, it makes little sense to make every security decision exclusively by querying an LLM again: Such approaches cause disproportionately high latency and costs, and are moreover often subject to the same structural vulnerabilities as the primary model itself.

Purely text-based detection (signatures, lists of known attacks, or classifying models) often fails in practice even with simple reformulations – natural language is too variable for that. This is precisely why many approaches unintentionally end up in a race against ever-new prompt formulations.

LINK2AI.Trust – An Open Platform

Various analysis modules evaluate an interaction from different perspectives as needed. LINK2AI.Trust can check user input for sensitive data / secrets and prevent transmission to the LLM. It performs security checks on the input (e.g., prompt injections), but also intensively examines the output – because successful attacks, i.e., those the LLM has fallen victim to, are the biggest problem. And LINK2AI.Trust verifies alignment with your rules and guardrails – and turns this into reliable quality assessments.

List of modules:

- **Prompt Injection Detection - Detection of direct and indirect attacks and manipulations**
- **Instruction Adherence - Analysis of the extent to which the language model has followed system instructions**
- **Jailbreak Detection - Detection of attempts to circumvent the global alignment of the language model**
- **General Request Metrics - Tracking of all interactions and key KPIs**

Looking Inside the Model: Attention and Trust Score

A particular strength of LINK2AI.Trust lies in the use of attention-based analysis methods. These allow us to look inside the model's black box, so to speak. In LINK2AI.Trust, the prompt is divided into a Trusted Part (e.g., system instructions and policies) and an Untrusted Part (e.g., user inputs, external documents). From the so-called attention signals, it can be derived how strongly the Trusted Part has shaped the output. This produces a compact Trust Score that indicates whether the application is "on track" – that is, remains aligned with the parts of the system instructions that describe the application's task as well as desired and undesired behavior – or whether an attack or misalignment has taken control.

Transformer models, on which LLMs are based, offer a built-in mechanism through self-attention to analyze which parts of the input were relevant for the output.

The practical advantage: This approach does not rely on known attack texts. In evaluation, it proved significantly more robust against rephrased prompt injection variants than common text-based detectors, for example. At the same time, this approach is considerably faster and more resource-efficient than querying an LLM. This makes it ideally suited as a first, rapid verification stage to detect suspicious interactions early and only trigger more elaborate verification mechanisms in justified cases.

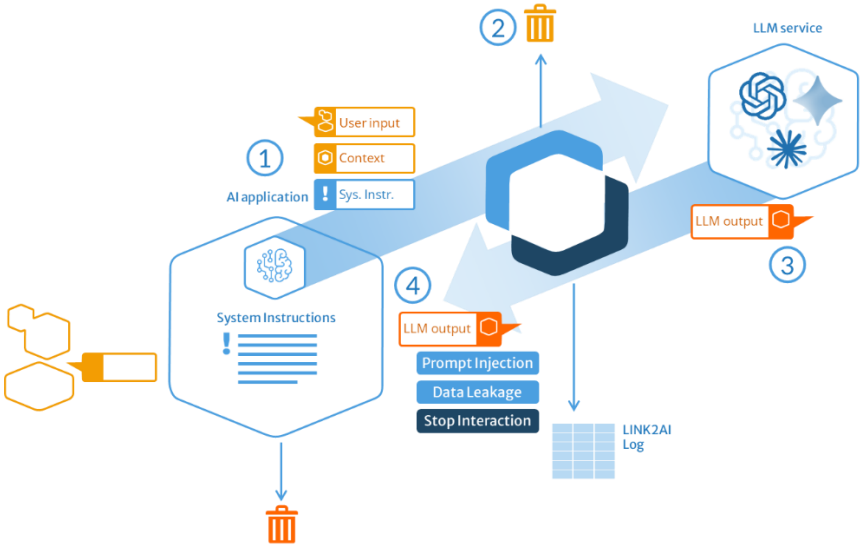
Finally, this approach is also suitable for detecting LLM errors beyond attacks. If a system instruction states, for example, "do not talk about competitor products," but the response does so anyway, this is not a security incident, yet it still shows up in the attention values. In this way, LINK2AI.Trust creates the foundation for secure, traceable, and continuously improvable LLM applications.

Production Use: Also Works with Cloud LLMs

In many production setups, the required model-internal signals of a proprietary cloud LLM are not directly accessible. LINK2AI.Trust has a pragmatic solution for this: A secondary, smaller LLM (control model) can be used to simulate the missing attention signals from the generated response of the "main LLM." This way, the approach remains applicable even when the "main LLM" is a black box.

Implementation

The process flow looks as follows:



1. Build request – The application creates the LLM request from user input, context (e.g., documents, etc.), and system instructions. Input and context can also be harmful or contain secrets.

2. Input check – LINK2AI.Trust verifies whether content must not be processed by the LLM service according to defined rules (e.g., PII, passwords, tokens, etc.). If that is case, transmission is blocked and a corresponding error code is returned to the application.

3. Output check – The LLM generates the output, which is evaluated by LINK2AI.Trust for safety and instruction adherence/quality.

4. Return with evaluation – LINK2AI.Trust returns the output to the application, including a safety and quality score. The decision whether to block the response is up to the application

LINK2AI.Trust can be integrated into an LLM application in such a way that inputs and/or outputs are verified without restructuring the application's fundamental logic. In practice, there are two typical patterns: Either LINK2AI.Trust is placed as a proxy in front of the actual LLM call, or the application calls LINK2AI.Trust as a separate service –

specifically for input, output, or both. Both variants pursue the same goal: detect risks early, handle them cleanly in case of emergency, and at the same time provide measurable signals to systematically evolve the application.

In terms of the process flow, this means: The application creates the context for the LLM call as usual (user input, system instructions, possibly documents or other sources). An input check can already take place at this point. Subsequently, the request is sent to the LLM, the response is generated, and then the output is evaluated by LINK2AI.Trust. This evaluation is returned to the application together with the output, so that the application itself can decide how to react.

The input check is primarily about clear "hard stops": If there are instructions or guardrails that certain information must not be sent to an LLM – such as personal data, passwords, tokens, or other secrets – LINK2AI.Trust can block the transmission and return an error code to the application. The application can then, for example, display an understandable error message, redact content and retry, or hand the process over to an alternative workflow. The key point is: unwanted processing of sensitive information should not only be noticed "after the fact," but should never be sent to the LLM service in the first place.

The output check evaluates two dimensions that should deliberately be treated differently in operation. First, the safety perspective: Does the response violate safety instructions or contain potentially harmful content? Such cases are typically "acute" and should lead to an immediate reaction from the LLM application (e.g., block, defuse, escalate). Second, the quality or adherence perspective: Does the model adhere to the system instructions and the application's "business rules"? If a system instruction states, for example, "Do not talk about competitor products" and the response nevertheless addresses competitor products, this is not a classic security incident – but it is poor quality that can impair user experience, brand, or process integrity.

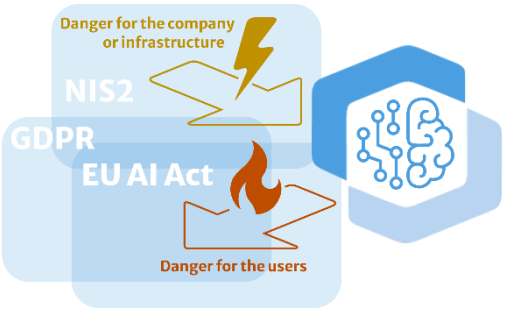
LINK2AI.Trust returns a Safety Score and a Quality Score together with the LLM output for the interaction with the language model. How these signals are handled is deliberately left in the hands of the LLM application: from blocking to redacting to retry or human review.

All interactions are stored together with the analysis results in a log data repository. This is important for operations for two reasons: First, safety-relevant anomalies can be traced and treated as incidents. Second – and often even more valuable – the quality signals enable continuous improvement of the application. While poor safety scores typically require immediate intervention, deviations in individual system instructions are often indications of which rules are formulated unclearly, which contexts are problematic, or which cases should be better tested in the future. This transforms "the AI is inconsistent" into a traceable, data-driven improvement process across prompts, updates, and releases.

For many teams, the easiest entry point is the proxy mode: The LLM endpoint is accessed via LINK2AI.Trust, authentication is additionally handled via a LINK2AI API token, and the response contains structured

analysis results alongside the LLM output. Alternatively, the service mode offers maximum control, for example when input checks must take place before the LLM call or when the application wants to trigger different paths depending on the result. Currently, LINK2AI.Trust supports text-based interactions; streaming is not yet covered in this version.

Compliance



With the EU AI Act and other European as well as national regulations, the requirements for the use of AI are increasing noticeably. Many companies are uncertain which regulations apply when; and what technical and organizational obligations arise from them for providers, integrators, and operators. One thing is clear: Without a robust compliance strategy, the productive operation of AI solutions will not be sustainably possible in many areas.

It is important to note that in practice, a large part of the obligations boils down to recurring core questions: Is the system controllable? Is its behavior traceable? Are security and data protection risks being controlled? And can all of this be demonstrated? This is exactly where LINK2AI.Trust comes in. The platform specifically supports the technical measures that are typically required in various regulatory frameworks – depending on role and risk class: continuous quality monitoring, protection against manipulation, enforcement of requirements (policies/guardrails), as well as logging and auditability in operation.

Depending on the use case, different regulatory frameworks may become relevant. The EU AI Act addresses, among other things, risk-based requirements, governance, and demonstrability. The GDPR requires protection of personal data and principles such as data minimization and purpose limitation. NIS2 tightens requirements for cybersecurity and risk management in many companies and supply chains. The Cyber Resilience Act (CRA) strengthens obligations for the secure development and operation of digital products. LINK2AI.Trust does not replace legal assessment – but it helps to consistently implement the technical requirements derived from it.

In practical terms, this means: LINK2AI.Trust can already prevent sensitive information (e.g., personal data, passwords, tokens, internal secrets) from being unintentionally transmitted to an LLM at the input stage. On the output side, the platform assesses whether responses violate safety requirements or internal policies; and whether the model reliably adheres to system instructions. Both are central to controlling risks in operation and securing the use of LLMs in regulated or reputation-critical scenarios.

Another compliance-relevant component is traceability in operation: LINK2AI.Trust stores interactions together with analysis results in a log data repository. This creates reliable information about how the application behaves over time, where deviations occur, and which measures are effective. This supports both operational processes

(incident handling) and the continuous improvement process that is implicitly or explicitly required by many regulations.

For companies deploying GenAI productively, the pragmatic approach is: first clarify which regulatory frameworks and roles (provider / operator / integrator) are relevant, then factor in the technical controls from the start, rather than "bolting them on" later. LINK2AI.Trust provides the technical foundation for this: controllable quality, secured interactions, and traceable evidence that can be utilized throughout the entire lifecycle of an AI application.