VALIDATION FRAMEWORK

‘Responsible use of Large Language Models (LLMs) for public information provision’

Table of Contents

Click on the chapter to navigate there directly

To make use of the functionalities in this interactive PDF, it is best to make use of a PDF viewer, such as Adobe Reader.

- Organisational measures
- Application design
- Evaluation
- Evaluation moments
- Additional information

This project was made possible by:

SIDN fonds

TOPSECTOR ICT dutch digital

Validation framework 'Responsible use of Large Language Models (LLMs) for public information provision'

This validation framework supports organisations in the responsible use of Large Language Model (LLM) applications for public information provision. AI models, such as GPT-5.2, Llama and GPT-NL, offer opportunities in making information widely available to citizens, organisations and civil servants. However, these opportunities are accompanied by risks. Therefore, it is necessary to verify that the technology functions properly before deploying it. This validation framework assists in the process of assessing and verifying the functioning of the LLM application.

What is public information provision?

Enabling that citizens, organisations and civil servants have access to reliable, understandable and accessible information about public services, policies, legislation, and regulations.

Terms in *italics* are described in the glossary on p. 52-53.

Why a validation framework?

Providing public information is a core task of government institutions. Ethical safeguards are an integral part of this, and this does not change when LLMs are introduced to this task.

LLMs are a form of generative AI. Unliked traditional AI applications, such as classification algorithms, both the type of input and output of an LLM can vary. This flexibility offers new possibilities for automated data processing. At the same time, it also entails unique risks, for example with regard to end-user privacy or the generation of inappropriate content. This also makes evaluating LLM applications a challenging task.

When considering the use of an LLM, the societal consequences must be taken into account. For example, using an LLM requires a lot of computing power, energy and data. As only few commercial parties are capable of offering this, the use of LLMs often entails dependencies. In addition, there are often copyright issues or questions regarding training data and its obtainment. In some cases the data has been collected or labelled by data workers whose labour rights are not fully respected.

Core principles

The following core principles are taken as a starting point for the responsible use of LLMs:

- **Effective:** the LLM application must have a clear purpose and is suitable for achieving this purpose.
- **Risk management:** Context-specific risk management measures, such as *guardrails* and *prompting*, provide quality assurance and mitigate risks.
- **Structural evaluation:** The quality and risks of the LLM application are structurally tested and monitored.
- **Transparency:** Interactions with the LLM provide a transparent user experience.
- **Interoperability:** Dependence on commercial LLMs and (American) cloud platforms is minimised with a view to European sovereignty.
- **Environmental impact:** The impact of the LLM on people and the natural environment is assessed and taken into account when deciding whether to use the technology.

Reading guide

Structure and scope of the validation framework

The validation framework describes best practices for 12 different dimensions regarding responsible use of LLMs in supporting public information provision. The best practices are based on practical experiences and scientific insights which are divided into three sections: Organisational measures, Application design and Evaluation. In the first two sections, the best practices are grouped thematically. In the Evaluation section, risks are first identified, followed by a thematic description of control measures for evaluating both individual components and the LLM applications as a whole. The framework is supplemented with relevant information on, among other things, the AI Act and specific evaluation methods. This framework does not provide an exhaustive list of safeguards that guarantee the responsible deployment of LLMs, but is a tool for development teams to assess how certain risks associated with the use of LLMs may be managed. Aspects relating to the information security of LLM applications fall outside the scope of this framework.

Target audience

This validation framework is intended for **multidisciplinary teams** within governmental organisations which are involved in the (potential) development of an LLM application for public information provision. The framework addresses concerns for both **management** and **technical development teams**. The validation framework requires input from and close cooperation with **stakeholders**, **end users**, and **domain experts** in order to deploy risk mitigation measures successfully.

The field of LLM applications is developing rapidly. Best practices described in this framework regarding control measures and evaluations must therefore be continuously reviewed in light of new insights.

LLM application development process

The development of an LLM application for public information provision follows a generic development process. This process, which is also followed for other software and AI applications, is elaborated here for the development of an LLM application. For the sake of readability, the development process is presented in a linear fashion. In practice, steps are often carried out iteratively. For example, it may become apparent in Step 3 that changes are needed in Steps 1 and 2.

- Organisational measures
- Application design
- Evaluation
- Evaluation moments
- Additional information

1 Problem definition

Start by determining the **Objective**: is an LLM really necessary, or would a simpler alternative suffice? At the same time, **Governance** is being established. This includes drawing up algorithm management policy with clear roles and responsibilities and an exit strategy.

2 Design

Define the **Objective** and the user requirements. The technical blueprint is established as part of the **Application Design**, during which the necessary (LLM) components and their interactions are determined. The **User interface** is designed in such a way that users understand that they are interacting with AI and can always find human support. During this process, **Privacy** safeguards are implemented and requirements from the AI Act are complied with (see p. 49-51).

3 Model engineering and product development

Select a **Large Language Model** that fits the defined requirements. Design the **System prompt** iteratively such that the LLM is used in a targeted manner. Identified risks are managed using **Guardrails**, which control, among other things, the input and output to and from the LLM. If the LLM application makes use of a search functionality, a **Knowledge base and search strategy** must be determined.

4 Testing and validation

Conduct a **Pre-production evaluation** before the LLM application is deployed. This evaluation differs from the evaluation of other AI systems. Individual components of the LLM application, as well as the application as a whole, are evaluated. Evaluation requires, among other things, context-specific evaluation datasets and thorough evaluation by domain experts and end users to determine the performance and impact of developed application.

5 Deployment and monitoring

Once an LLM application has been put into use, continuous **Monitoring** of the application is required. The effectiveness of the **Guardrails** must be tracked and the reliability of the entire system must be verified using measurable criteria.

6 Maintenance and iteration

Based on **Production evaluation** and **Monitoring**, the **System prompt**, **Guardrails** and/or components of the LLM application are adjusted if necessary. Whenever there is a change in the choice of the **Large Language Model** or the **Knowledge base and search strategy**, the **Pre-production evaluation** is repeated to ensure established (performance) requirements are met and to verify that the established frameworks for the **Objective** and **Governance** are being adhered to.

Life cycle inspired by the life cycle from ISO/IEC 22989:2022

Basic principles for evaluating LLM applications

Thorough evaluation is central to this validation framework. Responsible deployment of an LLM application requires that all individual components as well as the application as a whole have been carefully designed and evaluated. The following pages provide background information related to evaluating an LLM application. The Evaluation section discusses evaluation methods in more detail as part of a generally applicable step-by-step evaluation approach.

What does evaluation of an LLM application entail?

Evaluation is the systematic assessment of the LLM application's performance, for example in terms of accuracy, reliability, usefulness, safety, and efficiency, against established performance requirements. Choosing a suitable evaluation method is dependent on the type of task performed by the LLM application, the LLM components used, the impact and output of the LLM application, and the availability of a *benchmark dataset*. Each element, including user testing, stress testing and red teaming, is explained below.

Establishing performance requirements and performance metrics

Each performance requirement must be evaluated using carefully defined *performance metrics*. In the context of public information provision, the following performance requirements may be relevant: the output is based on relevant sources, the summary is accurate and complete, and the response is understandable for the intended end user. Performance can only be measured when it is precisely defined what a performance metric aims to measure. For example: what exactly is meant by a 'correct' summary or and 'understandable' response?

Evaluating LLM components

A suitable evaluation method depends on the task performed by an LLM component. Tasks performed by LLM components can be roughly divided into two categories:

- ***Classification* or *prediction*:** An LLM-driven privacy-guardrail for masking personal data (PII masking) is a classification task: does a piece of text contain personal data or not? (see also PV)
- **Generation:** Some LLM components generate content, such as textual responses in a chat. Evaluating generative tasks is complex because generated text can vary almost infinitely.

For the evaluation, it is irrelevant whether classification is implemented by means of a classification model or prompting.

Basic principles for evaluating LLM applications

Evaluating using benchmark datasets

A *benchmark dataset* is a carefully selected collection of test situations that supports the evaluation of an LLM component regarding a specific objective. Test situations are relevant and realistic for the specific context in which the LLM application is used. For each situation in the benchmark dataset, a *performance metric* is used to determine whether the LLM component handles the situation appropriately. Based on this score, performance of different LLMs can be compared with each other. Examples of performance metrics for LLM applications are accuracy, precision and recall. There are different types of benchmark datasets:

- **Situations with *ground truth labels*:** Classification and prediction tasks are evaluated using *ground truth labels*. These are labels determined by experts regarding the (in) correctness of a situation. Performance is determined by comparing the output of an LLM component with the ground truth label from the benchmark dataset. The LLM application is evaluated on the entire dataset using a performance metric. For example, the benchmark dataset of a privacy-guardrail that masks personal information consists of chat messages in which for each word in the chat message, it is indicated whether or not it is personal data. Depending on the type of application and the performance requirement to be measured, ground truth labels can for test situations can be annotated by experts, end-users, or non-experts (see box on page 6). Existing open-source benchmark datasets, such as BBQ, can also be used. However, context-specific benchmark datasets often need to be created to evaluate an LLM application. Once ground truth labels are known, the evaluation can also be performed automatically using CI/CD (see also PPE).
- **Situations without *ground truth labels*:** For generation tasks, a benchmark dataset usually consists of test situations without ground truth labels. This is the case, for example, when an LLM application summarises texts. A generated summary will almost always be different, so its quality must be assessed repeatedly. Nevertheless, it is still useful to work out sample results. This helps to define a suitable performance metric and can serve as an example for assessors. Evaluation can take place through assessment by experts, end-users, non-experts or by means of the LLM-as-a-judge evaluation method (see box on page 8).

User testing

Various types of testing can be used to test user interaction with a component of the LLM application or with the application as a whole.

- **Component-test:** Users are asked to assess a specific test situation for (a) specific performance metric(s). Component-testing does not require the application to be finished. It is possible that users may be presented with summaries and asked to assess their readability and correctness without interacting with the LLM application themselves.
- **Controlled experiment (A/B test):** During a controlled experiment, one or more variants (of a component in) the application are compared with each other to determine which variant performs best for a specific performance metric)
- **Bèta-test:** A beta-test is used to assess the user-friendliness and performance of the LLM application in a realistic setting. Through this test method, unexpected problems may unveil themselves which can be identified during production.

Basic principles for evaluating LLM applications

Stress testing, red-teaming and edge case testing

During evaluation, it is also necessary to test how the LLM application handles extreme scenarios. These are scenarios which fall outside the expected usage patterns, such as rare and/or controversial situations. Based on these tests, weaknesses, such as unexpected behaviour, in the LLM application can be identified. Separate datasets (with or without *ground truth labels*) can be used for stress testing or testing edge cases. This form of evaluation is usually referred to as *red-teaming* and is usually carried out by a specialist team to test for possible (systematic) abuse, such as *prompt injection*, *jailbreaking*, or other undesirable output.

Five ways to assess the output of an LLM application

1. **Expert evaluators:** The golden standard is evaluation by experts (experienced or domain experts). This evaluation method is very accurate, at the cost of being expensive and slow. In most cases, *ground truth labels* are created by expert evaluators.
2. **User assessment**: When the performance is linked to the usefulness of the application for regular users, it is important that outputs are assessed by users. Ensure that you have a diverse group of users who are collectively representative of the target group for the LLM application.
3. **Assessment by non-experts:** In some domains, it might be possible to have non-experts (such as the developers or external platform workers) assess the test situations. This method is often more accessible than using expert assessors or end-users, though also less accurate. If assessment by non-experts is used, for example in the initial start-up phase of a project, it must be considered whether this method should eventually be supplemented by expert or end-user assessments.
4. **Automatic comparison with *ground truth labels*:** When reliable data and labels are available, for example from benchmark datasets, the outputs of LLM components can be automatically compared with *ground truth labels*. This evaluation method is highly accurate and inexpensive when a quality dataset is available.
5. **LLM-as-a-judge:** In this evaluation method, another LLM is used to assess the output of an LLM component based on clear, predefined assessment criteria in a prompt. This is a highly scalable evaluation method, though less accurate. When using *LLM-as-a-judge*, it must first be determined whether the judging LLM is able to consistently and accurately evaluate the output of the generating LLM. This is done by testing the scores against specific performance requirements and measuring the degrees of agreement with human experts. In order to assess the generalisability of the results, it is advisable, as with regular data experiments, to split the dataset into a validation set and a test set.

Evaluation (E) p. 18

Step 1 Design evaluation process
Step 2 Benchmark dataset
Step 3 User testing
Step 4 Stress testing, red-teaming and edge case test

Objective (O) p. 10

- ☐ **O.1** Objectives and target audience of LLM application
  - **a.** Problem analysis
  - **b.** Organisational objectives and impact
  - **c.** Societal objectives and impact
  - **d.** Scope and target audience
- ☐ **O.2** Coordination with users
- ☐ **O.3** Suitable method
  - **a.** Alternatives
  - **b.** Comparison of methods
- ☐ **O.4** Identify domain-specific risks
- ☐ **O.5** Proportionality assessment

Large Language Model (LLM) p. 21

- ☐ **LLM.1** Choose the right type of model
- ☐ **LLM.2** Consider a local LLM if appropriate, or possible a parallel or externally hosted LLM
- ☐ **LLM.3** Adjust the chosen model through RAG or fine tuning
- ☐ **LLM.4** Have an exit strategy in place so that the used model can be easily replaced
- ☐ **LLM.5** Evaluate with users and domain experts (see E (p. 18))

Knowledge base and search strategy (KB) p. 35

- ☐ **KB.1** Identify suitable sources that help users meet information needs
- ☐ **KB.2** Determine appropriate search functionality
  - **a.** Determine how sources are chunked
  - **b.** Choose full text, semantic or hybrid search strategy
- ☐ **KB.3** Determine appropriate LLM for identified search functionality
- ☐ **KB.4** Evaluate functioning of knowledge base and search strategy (see E (p. 18))

Guardrails (GR) p. 39

- ☐ **GR.1** Design guardrails (risk management measure)
  - **a.** For user input to LLM application
  - **b.** For filtering personal information
  - **c.** For output from LLM application to end-user
  - **d.** For technical safeguards
- ☐ **GR.2** Test the designed guardrails (see E (p. 18))

Governance (GO) p. 13

- ☐ **GO.1** Management using current algorithm policy
- ☐ **GO.2** Carry out DPIA/IAMA
- ☐ **GO.3** Defining roles and responsibilities
  - **a.** During development/pilot
  - **b.** During use
- ☐ **GO.4** Exit-strategy for discontinuing LLM application

User interface (UI) p. 32

- ☐ **UI.1** Transparency regarding LLM interaction, model, scope and limitations
- ☐ **UI.2** Secure, accessible input using a proprietary front end with privacy by design
- ☐ **UI.3** Verifiability through source references and explanation of method
- ☐ **UI.4** Verify whether users read and understand information provided
- ☐ **UI.5** Easy access to human support
- ☐ **UI.6** Easy reporting of errors and visible follow-up
- ☐ **UI.7** Set up evaluation process (see E (p. 18))

Monitoring (M) p. 44

- ☐ **M.1** Keep constant track of performance and user interactions
  - **a.** Establish monitoring plan
  - **b.** Establish incident-handling strategy
  - **c.** Develop monitoring platform
- ☐ **M.2** Record operational performance and usage
- ☐ **M.3** Monitor use and effectiveness of applied guardrails
- ☐ **M.4** If desirable, monitor user interaction with LLM application
- ☐ **M.5** Assess how end-users experience user interface.

Production evaluation (PE) p. 47

- ☐ **PE.1** Short-term evaluation
  - **a.** Check whether objectives have been achieved
  - **b.** Determine user experience and confidence
  - **c.** Assess operational efficiency
- ☐ **PE.2** Long-term evaluation
  - **a.** Are core problems solved?
  - **b.** Are there any side effects or system impacts?
  - **c.** Are there safeguards for organisational and societal consequences?

Application design (AD) p. 15

- ☐ **AD.1** Information provision and scoping
- ☐ **AD.2** System architecture
  - **a.** Selecting relevant sources
  - **b.** Application function LLM
  - **c.** Quality and risk management by design
- ☐ **AD.3** Operation and costs

Pre-production evaluation (PPE) p. 42

- ☐ **PPE.1** Establish evaluation process (see E (p. 18))
- ☐ **PPE.2** Utilise CI/CD
- ☐ **PPE.3** Validate operation with stakeholders

System prompt (SP) p. 25

- ☐ **SP.1** Set up a detailed system prompt and identify contextual key concepts that are mentioned in the prompt
- ☐ **SP.2** Establish an evaluation process (see E (p. 18))
  - **a.** Make use of CI/CD

Privacy (PV) p. 29

- ☐ **PV.1** Lawful basis
- ☐ **PV.2** Data minimisation
- ☐ **PV.3** Prevent LLM provider accessing users' prompt- and chat history
- ☐ **PV.4** Prioritise own infrastructure
- ☐ **PV.5** Verify legal compliance with legal experts

Objective (O)

Large Language Models (LLMs) are not always suitable for improving public information provision. Whether or not an LLM application is desirable must be decided for each context, given the intended Objective (O). Among other things, it must be examined why a simpler alternative, such as a traditional search solution or additional human support, is not a suitable solution. In order to consider suitability properly, both organisational and social consequences must be taken into account.

Best practices

#	Title	Description
O.1	**Objectives and target audience of LLM application**	Identifying the objectives, consequences and target audience of the LLM application, including underlying problem analysis. It could be such that the outcome of this step is that an LLM application is not the appropriate solution for the identified problem
a	Problem analysis	▪ Identify the problem that the LLM application should help solve ▪ Analyse the severity of the problem (quantify, if possible) ▪ Describe the causes of the problem
b	Organisational goals and consequences	▪ Specify which core task of the organisation the LLM application supports. For example, facilitating understandable explanations of the law, legislation or policy processes. ▪ Describe the purpose and the desired effect of the LLM application on the organisation and its work processes (quantify if possible). ▪ Ensure that employees' knowledge regarding work processes is maintained.
c	Societal objectives and consequences	▪ Specify the societal objectives of the application. These could include improving access to information for citizens and organisations. ▪ Consider the possible (indirect or unintended) consequences of using an LLM application. These could include reduced trust in government, energy consumption and environmental impact, dependence on commercial parties and the impact of harmful data collection practices (e.g., with regards to IP or content moderation) ▪ Consider whether the LLM application is used for a reputationally sensitive government task (such as contact with citizens) or less sensitive tasks (such as internal usage to explain policy to civil servants). ▪ If it is a reputationally sensitive government task, heightened attention must be given to quality assurance.

Objective (O)

#	Title	Description
d	Scope and target audience	▪ Describe the specific tasks of the LLM application in line with the objectives described in a-c (e.g., summarising, answering questions, rewriting in understandable language, etc.). ▪ Determine who will use the LLM application ▪ Determine the limits and the application's scope: identify tasks that fall outside the scope of the application **For some LLM applications for public services, for example, it is important that the application only provides factual information and does not give advice or make recommendations on how to act.**

O.2	**Coordination with users**	▪ Involve users to better understand their information needs. Specify users' information need (retrieving information, presenting information or other)
		▪ Ask users which requirements the provision tool must meet. Refine the data obtained in **O.1** based on this information.
		  ▪ For example, is a high level of information density desirable, or must complex topics be explained in simpler, understandable language?
		  ▪ How should sources be cited, e.g., in running text or in a footnote?
		▪ Select domain experts who will serve as a feedback group in the (potential) development process of the LLM application (see **KB.3** and **PPE.2**).

O.3	**Suitable method**	Check whether the LLM application is a suitable method for the purpose and target audience as identified in **O.1a**.

a	Identify alternatives	▪ Identify alternative methods for the identified objectives. These should include at least the following:
		  ▪ Contact with an employee (by phone, chat, in person)
		  ▪ Improved (traditional) search functionality
		  ▪ Rule-based logic, decision trees, selection menus, manuals and/or FAQs
		  ▪ Manually written summaries or static summaries produced using LLMs, manually checked and published
		  ▪ (Q&A-)forum

b	Compare suitable methods	▪ Substantiate whether an LLM application, compared to identified alternatives, is a suitable method of achieving the identified objective. In doing so, compare at least: ▪ Effectiveness, risks (see also **O.4**), costs, development time and maintenance

Objective (O)

#	Title	Description

O.4	**Domain-specific risks**	If an LLM application has been identified as a suitable method in **O.3**:
		▪ Identify risks associated with using the LLM application in their specific context. Consider at least the following:
		  ▪ The factuality of generated output (e.g., hallucinations, non-existent sources)
		  ▪ Privacy (e.g., legal basis for data processing, handling of personal data)
		  ▪ Unappropriate use of the application (e.g., aggressive language, use for unintended tasks, *jailbreaking)*
		  ▪ Representativeness and bias in generated output (e.g., stigmatising content, unequal treatment)
		▪ Assess the likelihood of risks occurring and determine their impact.
		
		The impact of risks depends on the objectives of the application. For example, privacy safeguards are more important when personal data is processed by an LLM or when there is inquired into personal circumstances (because in this case the user could share personal data without being asked to). When the LLM application is solely used to retrieve information from a knowledge base, privacy may play a less important role.

O.5	**Proportionality**	Use the information obtained in **O.1-O.4** to assess whether it is proportionate to start the development of an LLM application: ▪ Weigh up all the identified disadvantages for the organisation, society and stakeholders against the extent to which the intended objective will be achieved by using the LLM application. ▪ Consider whether there are any (simpler) alternatives that could be used to achieve the identified objectives or similar objectives.

Please note: the outcome of this step may be such that the use of an LLM application is not suitable for the identified objective.

Governance (GO)

For the responsible use of an LLM application, a Governance strategy (GO) must been established. The application is managed in accordance with existing algorithm management policies. These policy define, among other things, the roles and responsibilities of the development team and an administrator. It also includes process descriptions for review and decision-making moments. In addition, an exit strategy has been outlined.

Best practices

#	Title	Description
GO.1	**Algorithm management policy**	▪ Fit the application to the organisation's existing algorithm policy ▪ When the application is put into use: publish the LLM application in the national Algorithm Register Inspiration for algorithm management policy can be found in the Algorithm Framework of the Ministry of the Interior and Kingdom Relations, the Code for Good Digital Public Administration (CODIO) for national government, and the AI Governance Framework of the Association of Dutch Municipalities (VNG).

GO.2	**DPIA and IAMA**	▪ Perform a risk classification for the LLM application. If there is an impact on the parties involved, perform a (pre-)IAMA:
		  ▪ Include the information collected from **O.1-O.5**
		  ▪ When the LLM application is used for a high-risk application as specified in the AI Act, the government agency must carry out the Fundamental Rights Impact Assessment (FRIA) in compliance with the AI Act. For more information, see page 29.
		▪ If desired or required: carry out a DPIA. As long as the use of LLM applications is new and innovative for the organisation, carrying out a DPIA is recommended.
		  ▪ Check whether a DPIA must be carried out in accordance with internal policy or in accordance with the criteria published by the Dutch Data Protection Authority (AP).
		  ▪ Further requirements relating to privacy are included in the PV dimension.

GO.3	**Roles and responsibilities**	▪ Defining roles and responsibilities within the development and management team Spend extra time and effort on this step if the LLM application is procured.

Governance (GO)

#	Title	Description

a	**During development/pilot**	▪ Agree on the final responsibility for the functioning of the LLM application with directors
		▪ Appoint the person that will be responsible for analysing performance metrics, user chat history (if decided to do so, see **PV.3** for more), and deciding on implementation after pre-production evaluation (**PPE**).
		▪ Involve domain experts and users in the development and evaluation of the LLM application:
		  ▪ To determine the information needs of target audience (see **O.2**)
		  ▪ To set up the **Knowledge base and search strategy (KB)**, draw up **Guardrails (GR)** and **System prompt (SP)**, carry out **Pre-production evaluation (PPE)** and **Production evaluation (PE)**

b	**During use**	▪ Discuss which responsibilities are assigned to directors, managers or the development team. Such as:
		  ▪ Monitoring and communicating performance and (new) risks in accordance with established objectives, as specified in **O.1**, for example through regular reporting.
		  ▪ (Public) accountability of the used technology.

GO.4	**Exit strategy**	▪ Develop an actionable plan if the AI model, or the entire application can no longer be deployed.

Application Design (AD)

The Application Design (AD) focuses on translating the objectives and risks into design choices for the architecture and overall functioning of the LLM application. This step also involves selecting the technical application pattern for LLM components, such as chat, *guardrails*, *Retrieval Augmented Generation (RAG)* or agents. These choices come together in the system architecture, which describes how all components (data, models, APIs) interact with each other.

Best practices

#	Title	Description
AD.1	**Information provision process and demarca-tion**	▪ Describe the intended information provision process and the role of the LLM application within this process (see **O**). In any case, describe what information will be collected and processed and consider interaction with external systems. ▪ Translate the objectives (defined in O) into technical functionalities and (sub)tasks for the application.
AD.2	**System architecture**	Determine which components the LLM application consists of and how they interact with each other. Consider both front-end and back-end components, including user interface, servers, (vector)databases, APIs. Functionalities, and LLM com-ponent(s) (either through an API service or locally hosted). Take the scalability and maintainability of the entire system into account.
a	Sources	▪ Determine which sources are used to meet users' information needs. When making this selection, take into account factors such as factual accuracy, quality, timeliness and the presence of personal data regarding these sources. ▪ Determine how these sources will be consulted technically. For example, through an application-specific vector database or via API to an existing, possibly external, database.

Application Design (AD)

#	Title	Description

b	Functioning of LLM application	▪ Determine which functionalities or tasks are performed by an LLM component
		▪ For each LLM component, determine a suitable application pattern for performing the (sub)task. For example:
		  ▪ Chatbot: To ask the user for information requirements.
		  ▪ *Retrieval Augmented Generation*: For performing simple search tasks.
		  ▪ LLM agent: for executing more complex tasks which require planning and/or the use of tools.
		  ▪ If using an agent, determine how the agent's memory will be organised. In the context of public information provision, simple strategies will often suffice, though in some cases (e.g., multiple or long user interactions) external memory may be necessary.
		  ▪ Classification: For the implementation of guardrails or LLM-as-a-judge evaluation methods.

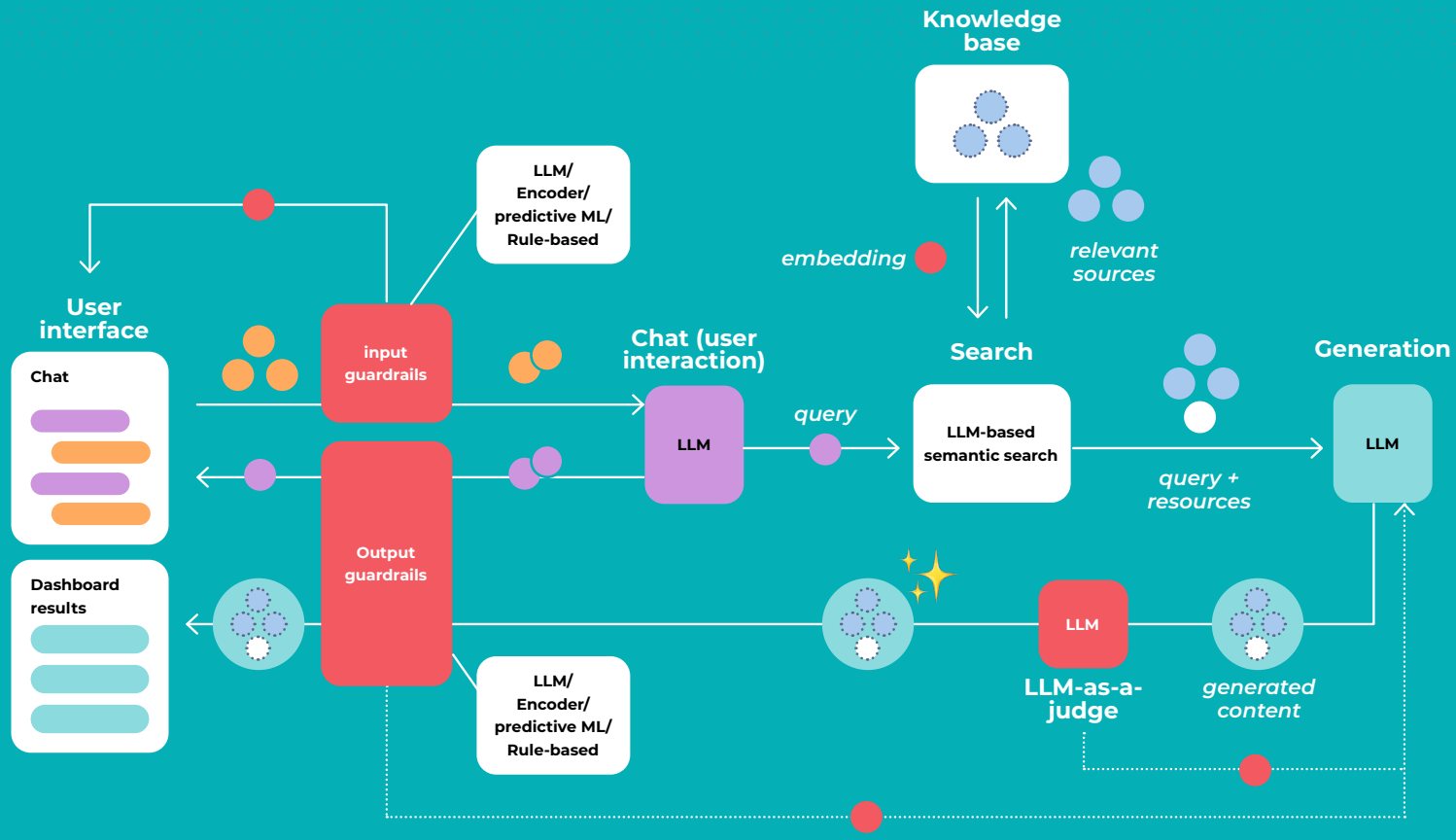
c	Quality and risk management by design	▪ For each component, determine which technical risk management measures will be used to guarantee quality and responsible deployment. Consider, in any case, the use of guardrails, system prompts, and LLM-as-a-judge. For more details see GR, SP, and E (p. 18).
AD.3	**Operation and costs**	▪ Estimate the costs for the necessary cloud resources, LLM API costs, data storage, and application development. ▪ Based on the (technical) feasibility, costs, objectives, and risks (see O), consider whether it is proportionate to continue developing the LLM application.

Example: voorRecht-rechtspraak

Application design

The voorRecht-rechtspraak LLM application consists out of various components. First, information is requested from the user via a chat interface. Based on the information retrieved, an LLM component sends an overview of the situation to the user for verification. If the user agrees with the outlined situation, a *RAG-application* is used to search relevant sources and summarises the relevant information from these sources. Various *guardrails* assess and filter the outputs.

LLMs are used in various stages of the application: in the chat and summary components, the search functionality and the *guardrails*.



voorRecht Rechtspraak

voorRecht-rechtspraak was developed by the Dutch judiciary in collaboration with T&T Data Consultancy and Deloitte.

User input
Chat component
Search (RAG)
Guardrails

Step-by-step plan for evaluating LLM applications

Evaluation is the core concept in this section and its associated dimensions. Building on the general introduction to evaluating LLMs (pages 6-8), this section explains a generally applicable step-by-step plan that can be used to evaluate LLM applications.

Step 1 – design the evaluation process

Design an evaluation methodology for each performance requirement of the LLM application. This may relate to one specific component or on a combination of components from the Application Design (AD). Take the following aspects into account:

- **LLM component and risk category:** Select a component from the LLM application and determine the risk category (such as: masking personal data, testing for bias or readability of output)
- **Evaluation method:** Determine whether a *benchmark dataset* or user test should be used. See p.8 and Steps 2 and 3.
- **Evaluator:** Determine who will evaluate the outputs of the LLM application (e.g., expert evaluators, end-users, non-experts, *LLM-as-a-judge*). See p.8.
- **Performance requirement:** Determine and describe the variable that is measured to assess the performance requirements (e.g., user satisfaction on a scale of 1 to 5)
- **Performance metric:** Determine how the performance requirement is quantified (e.g., accuracy, precision, (average) user satisfaction score).
- **Threshold value:** If relevant, determine an appropriate threshold value for the performance metric (e.g., "under normal use, *recall* is at least 0.85").

Step 2 – Benchmark dataset

Determine whether a customised benchmark dataset will be used. Establish requirement for the test situations and the corresponding judgements in the dataset used.

Step 2.1 Test situation:

Test situation must be sufficiently representative and diverse to reflect real-world scenarios. Take into account both intended and unintended use, as well as misuse. Various sources can be consulted to collect test situations:

- **Customised benchmark dataset:** An initial collection of test situations can often be compiled by the development team (non-experts). LLMs can provide support at this stage, for example by generating chat transcripts for a specific situation. To ensure representativeness and quality of the test situation, experts or (intended) end users must be involved in later stages.
- **Standard benchmark dataset:** In some cases, a standard benchmarking dataset, such as BBQ, can be used. In many cases, however, these are not specific enough for application-specific evaluation.

Pay extra attention to the representativeness of test situations if a performance requirement is linked to a threshold value under normal use. Avoid evaluation with perfect data in the benchmark dataset.

Step 2.2 Assessments:

- **Type of reviewer:** Test situations should preferably be labelled by (experienced or domain) experts. For simple tasks, non-experts can also perform the labelling. Make sure there is sufficient expertise and diversity between the human reviewers.
- **Define performance requirements:** Equip the reviewers with a clear definition of the performance requirements (e.g. relevance of found documents, presence of personal information)
- **Open-source data:** When choosing an existing benchmark dataset, assess whether the test situations and judgements can be trusted.

Step-by-step plan for evaluating LLM applications

- **LLM-as-a-judge:** When using LLM-as-a-judge, it must be validated whether the assessing LLM consistently and accurately assesses the generating LLM. For example, by comparing the scores with those of experts.

Step 2.3 Metadata:

Keep track of relevant metadata, such as the source of a test situation, the assessor and version of the LLM application.

Step 3 – User testing

- **Target audience:** Ensure that the group of users testing the interaction with the LLM application is diverse and representative of the target audience. See p. 8 for more details.
- **Task description:** For user testing, formulate a clear question and assignment. Focus on separating different performance requirements as much as possible.

Add relevant test situations to the benchmark dataset as prepared in Step 2. This will allow future changes to be tested, to ensure that these test situations are still being processed appropriately by the LLM application.

Step 4 – Stress testing, red-teaming and edge case testing

Monitoring: Incidents and complaints that come to light during **Monitoring (M)** or user testing can be added to the customised benchmark dataset.

Example evaluation methodologies for LLM application components

The table below provides several examples of a structured documentation of an LLM evaluation method. The rows show the different components of an LLM application that need to be evaluated. The columns represent the most important concepts from the evaluation step-by-step plan as detailed on pages 18-19. The table below is also referred to as an evaluation or risk taxonomy.

LLM-component	Risk category	Type of task	Test situation	Judgement	Assessment of test situation output	Ground truth labels	Source of ground truth labels	Metriek
Privacy guardrail	Privacy (masking personal data)	Classification	Chat messages	Present (yes/no)	Automatic comparison	Yes	Benchmark dataset/non-experts	FPR, FNR
Search function	Relevance	Classification, ranking or gen-eration	Search queries, relevant documents in knowledge base	Relevant (yes/no)	Automatic comparison	Yes	Expert assessors/ non-experts	Precision@k
RAG generation	Reliability	Generation	(Generated summary, origi-nal sources)	Contains information not found in the sources (yes/no)	Expert assessors, LLM as a Judge	No	N/A	FPR, FNR
RAG generation	Accessibility (readability)	Generation	(generated summary, original sources)	Scale 1-5	User, LLM-as-a-Judge	No	N/A	Average score

Large Language Model (LLM)

The chosen Large Language Model (LLM) largely determines the functionality and reliability of the component and the LLM application as a whole. There are many different types of LLM: ranging from large cloud-based models to small local models, whether commercial or open source. The chosen model must be in line with the intended Objective (O) and the formulated requirements surrounding Governance (GO) and Privacy (PV). LLMs are being developed at a rapid pace; remaining agile is therefore crucial. To avoid dependencies, the Application Design (AD) must be structured in such a way that the underlying models are relatively easy to exchange.

Identify

Objective: Determine the objectives of the LLM application in line with the objectives identified for the LLM application defined in, among others, O and AD. For LLMs, this involves striking a balance between performance, costs and control. Consider the following:

- **Performance:** The LLM must be able to perform the task accurately and within an acceptable time frame.
- Proportionality: The complexity of the model is proportional to the capabilities required for the intended task.
- **Governance:** Ensuring control over data processing.
- **Sustainability:** Consider the impact on people and the environment when choosing a specific large language model.

Risks: Determine which risks are relevant to the LLM components. Consider the risks identified in O for the LLM application. At a minimum, consider the following:

- **Interoperability:** The risk that the application becomes too dependent on one specific supplier or model architecture, making it difficult to switch to another LLM.
- **Confidentiality and IP:** The risk that the provider stores prompts and interactions with the API in order to train its own models.
- **Obsolescence:** Too rapid further development of the supplier, which can cause instability or unexpected output.

Large Language Model (LLM)

Evaluate

#	Title	Description
LLM.1	**Model type**	▪ Determine which type of model is suitable for the intended task. An LLM is not always necessary; consider other models as well (e.g., encoder-only models or more traditional predictive machine learning algorithms for classification or search tasks). ▪ The LLM may differ per task or component. Use a model that is as suitable as possible and has the right skills to perform adequately for the application (e.g., summarising, answering questions or reasoning). See also **O.1c**. Take into account at least skills, model size and energy consumption. **If an open-weights model is chosen, different techniques can be applied to make the model faster or smaller.**
LLM.2	**Hosting**	▪ Choose a model that meets the requirements for governance and privacy. See dimensions **GO** and **PV**. Weigh up the costs, performance and dependencies of self-hosted and externally hosted models: ▪ Self-hosted model: Offers maximum control over the processing of (special) personal data and other sensitive information that is not easy to mask. Self-hosted models also offer more scope for customisation and training with your own data. ▪ Externally hosted model: If an external model is chosen, try to stipulate contractually that the provider does not store or use system prompts, user prompts, or user interaction for training purposes.

Large Language Model (LLM)

#	Title	Description
LLM.3	**Model tuning**	▪ Tune the selected LLM for the desired application domain. Consider the following: ▪ *Retrieval Augmented Generation (RAG)*: If a proprietary domain-specific knowledge base is used (see KB). ▪ *Prompt engineering*: Adjusting the system prompt to improve the desired behaviour of the LLM application (see SP). ▪ *Fine-tuning*: Examples are used to teach the model what the desired behaviour is. There are various forms of fine tuning, including changing all weights in the model (full fine tuning), training a small part of the model or adding small modules. Well-known examples are Adapters and LoRA (Low-Rank Adaptation). These are forms of fine tuning that are fast and inexpensive. ▪ Model size: To make the model faster or smaller for use, techniques such as quantisation (reducing the precision of the weights in the model), pruning (removing unnecessary connections) and knowledge distillation (training a small model based on a larger model) can be applied. **As a general rule, the more the model's output needs to be directed towards a desired result, the greater the likelihood that the LLM will need to be fine-tuned. It is advisable to first investigate whether the set goals can be achieved with prompt engineering.**
LLM.4	**Exit strategy and interoperability**	▪ Set up the LLM application, infrastructure and test environment in such a way that, if desired, the chosen LLM can easily be replaced by another LLM. ▪ Check if this affects the effectiveness of the entire LLM application. See further dimensions Pre-production evaluation (**PPE**) and Monitoring (**M**). ▪ Check whether there are any dependencies with regard to underlying software.
LLM.5	**Evaluation process**	▪ Design and implement an evaluation method for each specific task and objective of the LLM (see E (p. 18)). ▪ Check with users and domain experts whether the LLM used performs sufficiently for the set performance requirements.

Example: GPT-NL

GPT-NL is a sovereign large language model developed specifically for the Dutch language and culture. It concerns only the large language model (foundation model) itself and not the AI system that may be built around it, such as ChatGPT for GPT-5. The training and fine-tuning are performed entirely in-house and have been carried out on Dutch infrastructure. The model has been developed using lawfully obtained data, without the use of data for which no permission has been granted by the rights holders. The data collection has been built from the ground up and has been checked for personal data, the protection of intellectual property and the exclusion of harmful content such as discriminatory or hateful texts. Authors of sources used receive a share of the proceeds from GPT-NL via a so-called 'content board'. The model is tailored to specific applications for which language models are often used, such as simplification, summarisation and RAG applications. In addition, GPT-NL focuses on developing and testing new and existing Dutch benchmark datasets, provides transparency about documentation and design choices in accordance with subsidy terms and conditions, and inspires future collaboration to create other generative AI solutions from Dutch soil.

GPT-NL

GPT-NL is being developed by TNO, SURF and the Netherlands Forensic Institute (NFI) as part of a sovereign Dutch large language model, with funding from the Netherlands Enterprise Agency (RVO) and the Ministry of Economic Affairs.

Most LLMs	GPT-NL
Use of scraped data without explicit consent	Legally obtained data
Limited control over copyright and privacy	Strict filters on personal data, IP and harmful content
Trained on infrastructure outside the Netherlands and/or Europe	Fully trained from scratch on Dutch infrastructure
Global, generic applications	Focused on public and social applications: RAG, summarisation, simplification
Generic linguistic and application-oriented benchmarks	LLM benchmarks for Dutch applications
Owned by international commercial parties, including governance	Content board for revenue sharing with rights holders
Little incentive for cooperation between national parties, e.g., for adjustments relating to local language and culture	Promoting cooperation between Dutch parties: ▪ Applications for Dutch public and private parties ▪ Subsequent generative AI initiatives
Disputed legal compliance (data protection, copyright-related, etc.)	Compliant with Dutch and European laws and regulations

System prompt (SP)

The system prompt forms the basis for interaction with an LLM. It guides how input, such as user chats or external sources, is processed. It defines desired output and boundaries and is therefore of great importance to the quality of the output. In addition, the System Prompt (SP) is, in addition to Guardrails (GR), an important measure to prevent incorrect, incomplete or undesirable output.

Identify

Objectives and risks: Determine the goals of the system prompt in line with the objectives and risk management of the LLM application defined elsewhere, including in O and AD, among other things. In any case, consider:

Intended use

- **Accuracy:** Correctly answering or handling relevant questions or topics that fall within the defined domain during normal use.
- **Factuality:** The system does not mention any non-existing references and does not add information that is not present in the original sources (*hallucinations*). References connect the correct source to the correct information.
- **Tone and style:** A consistent and appropriate tone and style of the output.
- **Structure:** An appropriate output structure for further processing (e.g. JSON).
- **Diversity:** The quality of the output is independent of the expected language variations of users (e.g., commonly used languages and dialects in Dutch society).
- **Desired interaction and inclusivity:** The output does not contain offensive, biased or otherwise undesirable texts.
- **Privacy:** Personal data of the user or third parties is not present in the output.

Unintended use and misuse

- **Scope:** The LLM application generates output on questions or topics that fall outside the defined application domain.
  - This also includes identifying cases that are too complex to be handled by the system.
- **Manipulation:** Detecting attempts relating to *prompt injection*, *jailbreaking* and other forms of system manipulation.
  - Examples include attempts to generate unwanted output, generate misinterpretations of laws and regulations that are more favourable to the user, or gain access to the system prompt, chat history or resources from the underlying knowledge base.

System prompt (SP)

Evaluate

#	Title	Description

SP.1	**Prompting technique**	▪ Write a detailed system prompt that includes all identified objectives. Make sure to consider at least the following:
		  ▪ Determine a suitable prompting structure for the identified objectives (e.g., Chain-Of-Thought, ReAct, Few-Shot prompting).
		  ▪ Write instructions for processing the context provided, such as transcripts of previous conversations or relevant documents from the knowledge base (see KB).
		  ▪ Instruct the LLM explicitly to indicate if the question asked cannot be answered based on the provided context or underlying sources. In the prompt, explicitly state to respond with: "I don't know" and that the LLM should not make up an answer.
		  ▪ When using an LLM agent, also determine instructions on how the large language model should communicate with external tools and services (e.g. via a Model Context Protocol server) and any externally stored memory.
		▪ Various strategies can aid in improving quality of outputs. Consider, for example, specific instructions for:
		  ▪ Tone and style: "be helpful", "respond in neutral terms"
		  ▪ Scope: "Limit yourself to practical solutions", "do not give legal advice"
		  ▪ Structure: "request substantiation or a step-by-step explanation", "provide information in JSON format" (consider using machine-readable output where possible)
		  ▪ Impersonation: "Assume the role of a professor"
		
		Writing a good (system) prompt requires many iterations. Keep in mind that best practices for prompting are constantly developing. Strategies that worked in the past may no longer work when a particular type of LLM is updated. Therefore, it is important to set up a robust evaluation process.

SP.2	**Evaluation process**	Even minor adjustments to a system prompt can lead to undesirable responses that were resolved in previous it-erations. It is therefore crucial to use a well-thought-out evaluation process. See E (p. 18).
a	CI/CD	▪ Integrate the evaluations into a CI/CD-workflow for the system prompt. This is crucial to quickly adapt the LLM application if necessary.

Example: system prompts

In general, there are several types of prompts used in an LLM application, such as for user interaction, for guardrails, and possibly for content generation and an LLM-as-a-judge evaluation process. A system prompt should be used for every task that utilises an LLM.

User interaction

system prompt:
You are a helpful assistant. Always collect the following information from the user first:
..

context - *chat history*
user: I have a question
chatbot: What is your question?

user prompt:
Can I sue my neighbour?

LLM

Before I can answer your question, I need some additional information. Do you own your home?

Guardrails

system prompt:
Verify whether the user's problems involves harassment, animal abuse, threats or violence. Do not address this and immediately refer to the police.

context - *chat history*
user: I have a question
chatbot: How nice! What is your question?

user prompt:
My neighbours leave their dog on the balcony all day in the heat without water. I am worried, what can I do?

LLM

{"value": TRUE, "filter_triggered": "animal_cruelty"}

Content generation

system prompt:
Answer the user's question based on the following sources. Link each sentence containing a fact to the source from which you obtained it.

Context - *sources*
Source A: Town hall opening hours: Monday: 7:30 a.m. – 5:30 p.m.
Source B: library opening hours: 8:00 a.m. – 8:00 p.m.

user prompt - *originele query*
What time does the town hall open on Monday?

LLM

The town hall opens on 7:30 on Monday [Source A].

LLM-as-a-judge

system prompt:
Assess whether the text in the generated output matches the given sources. Always provide your answer in JSON format.

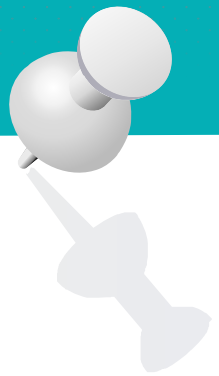
Context - *sources*
Source A: This book is about a boy who ...
Source B: This book was written in 2004

user prompt -
This book is about a boy who [...] and was written in 2005.

LLM

{"value":TRUE,
"errors":[{"sentence" "this book is about a boy and was written in 2005",
"source": "Source B"},
"source text": "this book was written in 2004"],}

Example: Wanted



Do you play a role in the development of an LLM application, and do you want to contribute to this Validation framework? Send us a letter containing approx. 250 words with regard to the 1) Objective and 2) Evaluation of your LLM application to info@algorithmaudit.eu

Privacy (PV)

The processing of personal data by LLM applications for the purpose of providing public information may sometimes be necessary. Personal data may, for example, be part of chat interactions, metadata or relevant sources. If personal data is processed in the LLM application, the GDPR must be complied with in order to guarantee the Privacy (PV) of users.

Identify

Objectives and risks: Determine objectives and risks relating to privacy, in line with the objectives identified in O. Because privacy is not a technical component with its own functionality, objectives and risks are dealt with together.

Identify the privacy risks that apply. Distinguish between personal data that is processed intentionally (such as IP address, cookies and/or metadata) and that which may be processed unintentionally (personal data in user interaction). Consider at least the following:

- **User privacy:**
  - Excessive collection of personal data, e.g., through metadata.
  - Users sharing personal data about themselves or others during interaction with the chat.
  - Use of cookies.
  - Personal data of users is shared with external parties such as suppliers.
- **Third-party privacy:**
  - Personal data contained in an organisation's internal data is shared with users or external parties, for example because this data is contained in the knowledge base.
- **Compliance with the GDPR:** The GDPR sets specific requirements for the processing of personal data, documentation and transparency. Also consider the various legal responsibilities of partners and suppliers for the LLM application used.

Privacy (PV)

Evaluate

#	Title	Description
PV.1	**GDPR, legal basis, purpose of processing and agreement**	▪ Document which personal data the LLM application processes. Distinguish between data that is processed intentionally (such as IP address or other metadata) and data that may be processed unintentionally (personal data in user interaction). ▪ Record the processing purposes and its legal basis. Please note: this may vary per personal data point. ▪ Conclude a processing agreement when multiple parties are involved in the processing of personal data (such as an external software supplier).

PV.2	**Data minimisation**	▪ Process as little personal data as necessary for the intended purpose. This means processing as little metadata as possible (such as IP addresses). If it is necessary to process this data, ensure that it is stored for as short a period as possible.
		▪ Ensure that personal data which a user unintentionally or unnecessarily shares with an LLM application, is processed as little as possible, preferably not at all. For example by:
		  ▪ Warning users not to share names, addresses and other personal information.
		  ▪ Masking personal data using content filters (see GR).

PV.3	**Access to user prompts and chat history**	▪ Configure the LLM application in such a way that the LLM provider does not have access to:
		  ▪ The system prompt of the LLM application.
		  ▪ The chat history of specific users and the collection of all user interaction with the LLM application, in-cluding metadata.
		▪ If possible, store chat history within the organisational infrastructure using the LLM application and check the re-tention period.
		
		Apply the 'least privilege' principle: users, systems and processors are only granted the minimum access rights neces-sary to perform their tasks in order to limit the risk of misuse or damage.

PV.4	**Prioritise own infrastructure**	▪ Configure the LLM application in such a way that as many components as possible are hosted on the organisation's own infrastructure.
PV.5	**Legal compliance**	▪ Check whether the measures taken sufficiently protect people's privacy and whether the GDPR is complied with, in line with the identified risks (for example in the DPIA, see also GO). Involve relevant privacy officers in this process.

Example: voorRecht-rechtspraak

Governance

The voorrecht-Rechtspraak application is managed in accordance with the algorithm management policy of the Dutch judiciary. This means that for the pilot and development phase, the team has made agreements with responsible managers about project planning, decision-making and selection moments. Since the pilot helps citizens online at all times with sharing information about legal conflicts, the application is monitored daily by an administrator for performance requirements, such as timeouts and activated content filters. Results and potential reputation risks are shared weekly by the development team with an administratively responsible team member.

During the development of the pilot, a risk assessment was carried out with a diverse group of stakeholders based on a simplified version of the Impact Assessment on Human Rights and Algorithms (IAMA). This showed that voorRecht-rechtspraak is a low-risk application and that it was not necessary to carry out a full IAMA. In addition, domain experts are continuously involved in the functioning of the *LLM application*, for example to evaluate how the RAG-application deals with borderline cases and to draw up and maintain a suitable *benchmark dataset*. The intellectual property rights have been purchased by the judiciary from supporting market parties.

voorRecht Rechtspraak

voorRecht-rechtspraak was developed by the Dutch judiciary in collaboration with T&T Data Consultancy and Deloitte.

Privacy

A Data Protection Impact Assessment (DPIA) was carried out to identify the risks of the pilot with experimental LLM technology in relation to data processing. This assessment explicitly states the basis for processing, the purposes of processing and the interests of those involved in data processing. To mask personal data from case law used in the knowledge base, proprietary masking scripts were written using Stanza. Azure content filters are used to detect hateful, intimidating and offensive content and to formulate an appropriate response. Similar Azure filters are used to protect the LLM application against *prompt injections*.

The judiciary has ruled that Microsoft does not have access to various system prompts from the LLM application and users' chat history with the chatbot. In order to effectively evaluate the pilot and continuously improve the LLM application, it has been decided to store users' chat history for a retention period of 60 days.

The tech stack of voorRecht-rechtspraak is designed in such a way that it is easy to switch between more privacy-friendly or European-oriented LLM providers. See LLM for more information.

User interface (UI)

Users typically only interact with an LLM through a User interface (UI). To provide users with a safe and pleasant experience (e.g., via a chatbot), it is important to carefully design the user interface and align it with user needs. For example, the interface must always make clear that users are interacting with an AI model. Additionally, public authorities must always offer human support in regard to their core tasks and responsibilities.

Identify

Objective: Define the goals for the user interface in line with the objectives defined in O and AD. Consider specific objectives and how these can be evaluated through relevant (user)tests (see E (p. 18)):

- **Accessibility and clarity:** The interface is sufficiently accessible and comprehensible for the intended users.
- **Transparency:** Make sure that it is visible the answer is generated by AI. Display details of the LLM; provide full source citations with reference date and link.
- **Feedback:** Include easy-to-find, low-threshold mechanisms for users to provide feedback in the user interface.

Risks: Determine which risks are relevant for the user interface. Consider the risks identified in O when the LLM application is in production. At minimum, consider:

- Deception due to unclear origin of information, reference dates, or indication of uncertainty.
- Privacy and security breaches through input, logging or rendering.
- Inaccessible or unclear UI (not WCAG compliant, overly complex language).
- Insufficient human safeguards or feedbackloops.

User interface (UI)

Evaluate

#	Title	Description
UI.1	**Introduction and expectations**	▪ Display a notice that responses are AI-generated. Explain the purpose and the limitations of the LLM application. Do not solely place the disclaimer in the footer of the user interface. Including such a notice is a requirement under the AI Act (see p.41-43) ▪ Display model details: model name, provider, version, date of latest update of the LLM application. ▪ In the context of the AI Act, stating the intended purpose of the LLM application and the purposes for which the LLM must not be used (with a view to unforeseen high-risk applications, see further p.41-43). ▪ Use plain B1-level language where possible. Avoid jargon and elaborate on complex concepts using explanatory tools.
UI.2	**Input and interaction**	▪ Use a custom *front-end*: the interaction takes place via a user interface managed by the organisation (no direct vendor-UI), with control over house style, logging, privacy by design and security. ▪ Comply demonstrably with *WCAG* 2.2 level AA for all critical user flows (keyboard navigation, focus visibility, contrast, semantic structure/ARIA, error messages in plain language). ▪ Warn users not to share any sensitive or personal data. Explain the principle of data minimisation (see PV.2)
UI.3	**Answer and transparency regarding technology and sources**	▪ Cite sources alongside responses. Consider: title, brief description of source publication date, and direct link. ▪ Provide a concise explanation of whether and how *RAG*, *fine-tuning* or *system prompts* are applied. State the reference date for time-sensitive information. ▪ Enable users to re-request sources for an answer or to export answers.
UI.4	**Verification**	▪ Evaluate whether users read and understand instructions, disclaimers and information provided.
UI.5	**Escalation to a human agent**	▪ Enable users to contact a human representative via a clearly visible option. State the communication channel, the availability of agents and the expected response time. ▪ Arrange contextual handover with the user's consent (data sharing), including status indication of the request.

User interface (UI)

#	Title	Description
UI.6	**Feedback and learning**	▪ Ask users for feedback, for example through 👍and 👎feedback buttons to report inaccurate, outdated, unclear, biased or harmful content. ▪ Users may also be asked to highlight specific passages to bring notable content to the attention of adminis-trators ▪ Inform the user on how their shared feedback is processed
UI.7	**Evaluation process**	▪ Design and implement an evaluation method for each specific objective of the user interface (see E (p. 18)). For most objectives, a user test is the most suitable method for evaluating the user interface. Often, multiple objectives can be combined into a single larger test. ▪ Determine in advance what is deemed “good” for the UI. For example, transparent (origin and sources clearly indicated), usefulness (key tasks can be completed smoothly), accessible (compliant with WCAG-guidelines), safe (no unintended data sharing or injections) and auditable (traceable source attribution). **Example tests include: task tests (find source/reference data), comprehension test of disclaimer/model info, and a UI-audit on consistent, accessible presentation**

Knowledge base and search strategy (KB)

The search functionality of an LLM application plays an important role in finding relevant sources to support public information provision. The quality of the search functionality depends on the quality of the Knowledge base and search strategy (KB).

Identify

Objective: Define goals relating to the knowledge back and search strategy in line with the objectives identified in O and AD. Consider:

- **Knowledge base:** Quality of sources (e.g., are texts verified for accuracy, representativeness and up-to-date).
- **Search functionality:** Relevance of retrieved sources.
- **Response generation:** Reliability of the responses generated by the LLM application.

Risks: Determine which risks are relevant for the knowledge base and search functionality. Consider the risks identified under O for the entirety of the LLM application. At minimum, consider:

- **Reliability:** Reference to non-existent sources and misrepresentation of information.
- **Privacy:** Sharing of personal data with the user via information retrieved from sources (data breach).
- **Bias:** Representativeness of sources used, which may lead to bias in generated output (e.g., stigmatization, unequal treatment).
- ***Prompt injection*:** User input through which instructions are used to mislead an LLM application into causing undesired behaviour.
- **Topicality:** Reference to outdated sources.

Knowledge base and search strategy (KB)

Evaluate

#	Title	Description
KB.1	**Suitable sources**	▪ Design evaluation criteria to assess the quality of sources against the established objectives. ▪ Determine which sources can be used as knowledge base to meet users' information needs. ▪ Evaluate whether the sources are of sufficient quality for the objectives of the component and the LLM application as a whole. See KB.4.
KB.2	**Search functionality**	There is no single perfect search strategy. The best strategy is context-dependent and differs per tasks for which the LLM application is deployed. Building a search functionality consists out of two main components: chunking documents and establishing a search strategy.

a	Chunking	▪ Determine to what extent documents need to be split (chunking). This depends on the length of the documents and how the chosen LLM handles large documents.
		▪ Rules of thumb for *chunking*:
		  ▪ If specific answers are desired: create large chunks of text. This generally works better because a targeted answer can be constructed from a broad context.
		  ▪ If the answer is likely to be composed from multiple documents or involves a broad search query: smaller chunks of text generally work better.
		
		In general: coarser segmentation provides more context but risks losing detail. Finer-grained segmentation focuses more on details, but overall context may be missing.
		
		Literature can aid in finding the right chunking strategy, such as fixed length, recursive token, and contextual chunking. See also Chroma's chunking strategies or Anthropic's contextual chunking.

Knowledge base and search strategy (KB)

#	Title	Description

b	Search strategy	▪ Determine which strategy is most suitable for the LLM application:
		  ▪ Full-text search: a strategy that searches for documents containing the same word as the query
		  ▪ Semantic search: a strategy that takes into account the contextual meaning of words, take into consideration the choice of model used to perform semantic search.
		  ▪ Hybrid search: combination of full-text and semantic search.
		▪ Formulate realistic queries to test the knowledge base. See also E (p. 18).
		  ▪ Enriching the query with context can lead to better results
		
		Various models used for semantic search can be found on HuggingFace.

KB.3	**LLM functionality**	▪ Choose a suitable model (see further under LLM) and an appropriate prompting strategy (see further under SP).
		▪ Determine how the LLM interacts with the knowledge base. Several options exist, such as (see also AD):
		  ▪ Based on the conversation in chat, a single query is generated, followed by the generation of a response using *Retrieval Augmented Generation (RAG)*.
		  ▪ Agents interact iteratively with the knowledge base to retrieve relevant information and ultimately generate a response.
		
		In protocols such as the Model Context Protocol (MCP), the manner in which AI models can securely and consistently exchange context (such as data, tools, and memory) with other systems can be specified.

KB.4	**Evaluation**	Evaluate the quality and performance of the knowledge base, search functionality and LLM functionality both as separate components and as a whole. Establish an evaluation process for assessing the established knowledge base and search strategy (see E (p. 18)).

Example: voorRecht-rechtspraak

Knowledge base and search strategy

During the development of voorRecht-rechtspraak, various experiments were conducted to automatically evaluate the search strategy for case law.

One of these experiments is described here. Based on 10.000 judicial rulings, questions were generated and submitted to the chatbot. For example, for a ruling concerning noise nuisance by neighbours, an LLM generated the question: “What can I do when my neighbour causes noise nuisance at night”? These questions were then submitted to the LLM application. Based on the sources retrieved, it was checked whether the document on which the question was based appeared in the top-k results and a performance metric was calculated. Using the performance metric, the effectiveness of the search strategy can be unambiguously compared against other search strategies.

Based on such an evaluation process, the best *chunking strategy* can be determined. In the case of voorRecht-rechtspraak, it was found that passages from multiple different sources are generally relevant, making it more effective to split sources into small segments. In addition to this insight, it was decided to add summaries of the full ruling to each chunk, so that the main thread of the ruling is preserved.

voorRecht Rechtspraak

voorRecht-rechtspraak was developed by the Dutch judiciary in collaboration with T&T Data Consultancy and Deloitte.

User tests and expert reviews

voorRecht-rechtspraak and its associated knowledge base and search strategy were tested by expert reviewers, such as judges and legal professionals, to assess the legal accuracy and applicability of the LLM application’s output.

During the user test, judges and legal practitioners submitted questions to the LLM application, after which they validated the search results and generated answers. These validations were carried out iteratively to verify whether the search strategy was effective.

The knowledge base and search strategy were subsequently tested with end users. During these tests, it was investigated to what extent citizens found the information presented to be useful.

All outcomes of the user tests were stored in a *benchmark dataset*, which can be used to (automatically) evaluate the LLM application when future changes are made.

Guardrails (GR)

Guardrails **(GR) are risk management measures for LLM applications. These measures are taken to deploy the technology as safely, reliably, and fairly as possible. Guardrails can, for example, check the input of LLM applications for the presence of personal data, or prevent the output from containing harmful information or irrelevant answers. Guardrails are necessary to manage the risks associated with the use of an LLM application.**

Identify

Objective and risks: Define objectives and risks for the guardrails, in line with the objectives for the LLM application as identified in O. Since guardrails are intended to manage risks, objectives and risks are addressed together.

Determine which risks can be managed through guardrails. At minimum, consider:

- **Undesirable use of the LLM application:**
  - Aggressive language.
  - Emergency situations: conversations involving self-harm, violence, or abuse.
  - Use for unintended tasks, jailbreaking, and *prompt injections*, whereby users attempt to make the model do things other than its intended purpose. This can occur directly (e.g., through chat interaction) or indirectly (e.g., through a document submitted alongside a request).
- **Privacy:** The sharing of personal data by the user with the LLM application. This refers to personal information in the chat interaction, not to deliberately processed (meta)data.

Guardrails (GR)

Evaluate

#	Title	Description
GR.1	**Design guardrails**	Determine an appropriate guardrail for each identified risk.

a	Input guardrails	Content filters are filters applied to the user input before it is shared with an LLM.
		▪ Design an appropriate guardrail for each identified risk. Existing libraries may be used for this purpose. Consider various options:
		  ▪ Hardcoded filters.
		  ▪ Specialised predictive models.
		  ▪ LLM-based guardrails.
		▪ If existing content filters are insufficient, design custom content filters.
		▪ Scope the LLM application and the underlying LLM via the *system prompt* (see SP), such that it only responds to user questions within the application's scope, as established under O.

b	Personal data	Identifying and, potentially, masking personal data is a specific type of guardrail, also referred to as Personally Identifiable Information (PII) masking. ▪ Determine what types of personal data users may (un)intentionally share with the LLM application. ▪ When designing the guardrail, consider whether the methods used are suited to Dutch language. ▪ Establish an acceptable error rate for output in which personal data has been incorrectly masked by the used content filters. The EPDB has issued guidance on how to prevent the indirect processing of personal data when using GPAI models (see paragraph 27-42)

Guardrails (GR)

#	Title	Description

c	Output guardrails	▪ Determine what constitutes undesirable output for the specific LLM application and calibrate content filters accordingly.
		▪ Establish an acceptable percentage of output containing undesirable content. Ensure that performance criteria are measurable.
		▪ Cite the sources on which the output of the LLM application is based:
		  ▪ Indicate on the basis of which passage from the underlying sources the output was generated.
		  ▪ Do not include information without a source in the output.
		▪ Establish an acceptable percentage of output containing incorrect information.
		
		Note: undesirable output is a broad concept and covers, among other things, stigmatising language misinformation and disinformation. Whereas correct information can be factually demonstrated to be wrong, undesirable content is less straightforward to classify.

d	Technical safeguards	▪ Detect automated interaction with the LLM application (bot-detection), for example through the use of a firewall. ▪ Set a (maximum) token limit during chat interaction. ▪ Only allow input which is necessary; for example, do not allow images or documents.
GR.2	**Test guardrails**	▪ Test each guardrail as implemented under GR.1 using an evaluation process. See E (p. 18). **For guardrails, evaluation using a benchmark dataset is typically preferred**

Pre-production evaluation (PPE)

Pre-production evaluation (PPE) focuses on testing the LLM application before it goes (back) into production. Setting up PPE can also assist in making the right choices during the development process, such as the choice of a specific LLM (LLM) or System prompt (SP).

Identify

Objective: Define goals relating to pre-production evaluation in line with the objectives identified in O and AD. At minimum, consider:

- **Validation of requirements:** Establishing that the LLM application meets the predefined safety, speed, and accuracy requirements.
- **Baseline:** Recording the performance of the first accepted version so that future updates can be benchmarked against it.
- **Reproducibility:** Reproducible results increase the reliability of findings.

Risks: Determine which risks are relevant for pre-production evaluation. Consider risks arising from inadequate evaluation or flawed evaluation:

- **Generalisability:** The risk that the LLM application performs well on evaluation data but underperforms in practice.
- **Subjectivity:** The risk that new versions are pushed to production based on intuition rather than systematic evaluation based on quantitative data.

Pre-production evaluation (PPE)

Evaluate

#	Title	Description
PPE.1	**Establish evaluation process**	▪ Establish an evaluation process, see E (p. 18).
PPE.2	**CI/CD**	Automatically test changes made to the LLM application using *CI/CD*.
a	Unit tests	▪ Write specific test cases for independent components of the LLM application so that they can be tested individually.
b	Integration tests	▪ When multiple components of the LLM application are interdependent, provide integration tests to verify that the application functions correctly in its entirety.
PPE.3	**Validate with stakeholders**	▪ Involve stakeholders in the evaluation of significant changes to the LLM application. Aim to strike balance between domain expertise and user experience ▪ Verify whether predefined performance requirements are met. For significant changes, always check whether these affect the performance of the LLM application.
a	Domain expert	▪ Evaluate generated content with domain experts to safeguard quality, accuracy, and applicability. Verify whether the domain-specific risks from **O.4** have been sufficiently mitigated.
b	End user	▪ Conduct user tests to measure whether the application works for end-users; see also E (p. 18).

Monitoring (M)

Once the LLM application is in production, continuous Monitoring (M) is required to safeguard operational performance, safety and quality. This includes oversight of the operational functioning of the application, implemented risk management measures and user interactions.

Identify

Purpose: Define monitoring objectives in line with the objectives defined under O and AD. Consider:

- **Validation of requirements:** Establishing that the LLM application meets the predefined safety, speed, and performance requirements.
- **Refining requirements:** Identifying areas for improvement based on user interactions.
- **Complaints and incidents:** Appropriate handling of complaints and incidents.

Risks: Determine the risks of (inadequate) monitoring. At minimum, consider:

- **Failure to meet performance requirements:** Not fulfilling performance requirements relating to privacy, quality, safety and operational performance.
- **Missing signals:** Inadequate incident response mechanism and/or exit strategy.

Monitoring (M)

Evaluate

#	Title	Description
M.1	**Monitoring strategy**	LLM applications require constant oversight of performance and user interactions.
a	Monitoring plan	▪ Record what (see **M.2-M.5**), when, by whom (see **GO.3**), and how (manually, automated) monitoring takes place. Plan at minimum continuous monitoring (daily or weekly) and fixed periodic evaluation moments (every quarter or (half) year).
b	Incident strategy	▪ Determine how complaints and incidents are handled, by whom (see **GO.3**) and within what timeframe. Ensure a clear link to the exit strategy (**GO.4**). ▪ Develop a clear complaints and incident reporting mechanism, whereby each incident is followed by an incident review, after which performance requirements and the evaluation process (as established on **E (p. 18)**) may be updated.
c	Monitoring platform	▪ Develop a monitoring platform that allows the application's performance to be monitored quickly and clearly
M.2	**Operational performance and usage**	▪ Monitor technical system performance, such as speed (latency), capacity (token usage), and resource utilisation (memory/servers). Also log (software) errors arising in the system.
M.3	**Monitor risk management**	▪ Monitor the guardrails (**GR**) applied to the LLM application. Watch the frequency of content filter triggers, (in)correct masking of personal data and context-specific tasks the application is not supposed to perform. ▪ Discuss over- and underperformance of the model with relevant stakeholders.
M.4	**User interaction with model**	▪ Where desirable, monitor the quality of the application based on analysis of chat transcripts, user feedback and automated evaluations. Identify areas for improvement with respect to quality and user-friendliness.
M.5	**User interaction with user interface**	▪ Monitor how users experience the user interface (**UI**). Analyse implicit behaviours: which features and buttons are used, where users drop off (drop-off points) and what the most common navigation paths are. Also collect explicit feedback (e.g., via a feedback form).

Example: voorRecht-rechtspraak

System prompt

The system prompt of voorRecht-rechtspraak is regularly updated based on monitoring findings. These changes are implemented, for example, to provide users with better information. To prevent a new update from reintroducing old issues, a *benchmark dataset* of test scenarios in which previous problems occurred has been compiled. This allows verification, after a system prompt update, that old problems do not recurre.

One example is the unnecessary repetition of a question the user has already answered during the conversation. This came to light in a chat about the painting of an apartment building by the owners' association (VvE). The chatbot asked whether the user lived in an apartment building, even though the user had already indicated this in an earlier message.

voorRecht Rechtspraak

voorRecht-rechtspraak was developed by the Dutch judiciary in collaboration with T&T Data Consultancy and Deloitte.

Evalueren en monitoren

For voorRecht-rechtspraak, a dashboard has been developed to monitor the use of the LLM application. This dashboard is only accessible to authorised staff (see GO.3).

In addition, the *LLM-as-a-judge* evaluation methodology is used to verify whether the output of the LLM application contains legal advice (which is not the intention), whether personal data is correctly masked and whether *hallucinations* occur in the output of the *RAG-application*. Low-scoring performance metrics are studied carefully by the development team. Deviating chats are first reviewed by domain experts to identify in which context the LLM application does not perform optimally. Relevant cases are included in the *benchmark dataset* so that these situations can be tested in future versions of the LLM application.

Production evaluation (PE)

During Production evaluation (PE), the real-world impact of the LLM application is assessed. Whereas Pre-Production Evaluation (PPE) and Monitoring (M) evaluate the technical performance of the model (does it work?), PE assesses the effectiveness of the solution (does it help?). This evaluation is benchmarked against the objectives and the problem analysis established in phase O.1.

Identify

Objective: Define goals for Production evaluation in line with the objectives identified in O and AD. Consider:

- **Problem-solving capacity:** Validating whether the core problem (e.g., information overload or long waiting times) is genuinely resolved by the LLM application.
- **Process and chain effects:** Making visible how the LLM application affects work processes and the customer experience (e.g., channel shift from telephone to digital or changes in processing times).
- **Accessibility and reach:** Verifying whether the solution reaches the intended target group, including those with limited digital literacy, and whether trust in the service is maintained.
- **Sustainability of the solution:** Evaluating whether the long-term benefits outweigh the structural management burden and costs.

Risks: Determine which risks arise from deploying the LLM application in a societal and organisational context. At minimum, consider:

- **Problem displacement:** The risk that the solution in one place causes unforeseen problems elsewhere in the chain.
- **Accessibility:** The risk that the LLM application unintentionally creates a barrier for specific groups, thereby increasing inequality in service delivery.
- **One-sided metrics:** The risk of steering on superficial figures (such as 'number of users') while the underlying problem (the actual need for assistance) remains unresolved.

Production evaluation (PE)

Evaluate

#	Title	Description
PE.1	**Short-term production evaluation**	Evaluate the direct, measurable impact of the LLM application in practice.
a	Objective realisation at application level	▪ Determine whether the individual information need or user problem is genuinely resolved through the use of the LLM application (see **O.1d**) ▪ Investigate whether the application leads to the desired follow-up action (e.g., the citizen understands the letter, the reader borrows the book, or the person seeking legal advice takes the correct legal step).
b	User experience and trust	▪ Determine whether the LLM application reaches the intended users (including those less digitally literate) or whether certain groups drop out.
c	Operation efficiency	▪ Investigate the effect of the LLM application on the workload of staff (e.g., fewer telephone calls, less or better preparation of case files). ▪ Investigate whether the processing time for citizens or professionals has been reduced compared to baseline (see **O.1b**).
PE.2	**Long-term production. evaluation**	Evaluate the strategic and structural impact of the LLM application over time.
a	Resolution of core problem	▪ Reflect on the problem analysis in **O.1a**. Has the quantified problem (e.g., excessive workload, incomprehensible information) been significantly reduced by the deployment of the LLM application? ▪ Investigate whether the target audience is shifting across channels (e.g., shift from physical encounter to digital service points, allowing experts more time for complex cases).
b	Side effects and system impact	▪ Determine whether the application has had unforeseen consequences elsewhere in the chain (e.g., does better information provision lead to more objections because citizens have become more assertive?). ▪ Investigate whether a cultural shift is visible in how citizens or staff approach the subject matter (e.g., a shift from conflict to dialogue).
c	Organisational embedding	▪ Determine whether the management costs (technical development, API costs, moderation) are proportionate to the social and operational cost in the long term. ▪ Investigate whether the solution is technically and organisationally sustainable: is the dependency on the vendor/technology too great (e.g., due to vendor lock-in)?

AI Act

The AI Act (AIA) imposes requirements on various types of AI systems and models. In most cases, the AI Act imposes no requirements on LLM applications for public information provision beyond transparency requirements. Nevertheless, it is important to determine which category under the AI Act the application falls into, so that legal requirements can be met where applicable.

Name	Example
General purpose AI model	GPT5.2 (OpenAI), Sonnet (Anthropic), GPT-NL (TNO, SURF, NFI et al.)
General purpose AI system	ChatGPT (OpenAI), Claude (Anthropic)
AI system	The 'explorer' part of voorRecht-rechtspraak (making use of a GPAI model)

#	Title	Description
AIA.1	**Transparency**	▪ For direct user interaction with an AI system (such as a chatbot): verify whether it is clearly stated that the user is interacting with an AI system. ▪ For content generation (e.g., summary or text): verify whether it is clearly stated that the content is AI-generated.
AIA.2	**General Purpose AI**	▪ Verify whether substantial modifications have been made to the underlying LLM. If so, seek (legal) counsel as to whether the organisation may be considered a developer of the model under the AI Act.
AIA.3	**High-risk AI system**	▪ Verify whether the application is deployed for a high-risk application.
AIA.4	**Intended purpose**	▪ For in-house development: verify that the user instructions and accompanying disclaimers clearly describe the intended purpose of the application. Ensure that it is also described for what purposes the application is not intended, in particular with regard to foreseeable misuse for a high-risk application. ▪ For procured systems: verify that the application is used in line with the intended purpose as established by the vendor. If used for a high-risk application that does not correspond to the application's intended purpose, seek (legal) counsel as to whether the organisation may be considered a developer of the model under the law. Example: "This application must not be used in the processes relating to the application, assessment, enforcement and supervision of, or recovery of, benefits, allowances, or essential public services."

AI act

1 Transparency requirements

It must always be clear to the user that they are interacting with an AI system. The developer must design the application in such a way that this is evident to the user.

If the LLM application is used to generate texts that are subsequently published as part of public information provision (e.g., as text on an informational website), it must be stated that the text was automatically generated.

See Article 50 of the AI Act for further information.

2 Requirements for GPAI Models

Legal framework

The AI Act imposes requirements on GPAI models. These are AI models capable of performing a wide range of different tasks. In the context of this framework, these are the LLMs used by the application (e.g., GPT-5 from OpenAI, GPT-NL, Sonnet from Anthropic, etc.). The AI Act imposes requirements only on the provider (supplier) of the GPAI model. In most cases, there are no requirements on the developer of an application that makes use of an existing GPAI model.

Note: In cases of significant modifications to the LLM whereby the capabilities or risks of the model are substantially altered, the developer making those modifications may be considered the provider (supplier) of the model under the law. This may have far-reaching consequences for the requirements imposed by the AI Act. Seek (legal) counsel prior to making any modifications to the model.

Available documentation

Providers of GPAI models must make documentation available. The following information must be made available:

- Technical documentation of the model, including training and testing processes and evaluation results.
- Documentation for other AI providers intending to integrate the GPAI model into an application, including details of its specifications and limitations.
- A detailed summary of the content and data used to train the AI model.
- An exception applies for open source models: these models are not required to make such documentation available.

This documentation can be used in the selection of a model (see LLM). As this is a statutory obligation, the absence of the above documentation means that the provider of a GPAI model does not comply with European legislation.

AI act

3 High-risk applications

The AI Act sets out extensive requirements for a specific category of AI systems: high-risk applications. High-risk applications are exhaustively listed in the AI Act. This means that if an application is not used for one of the listed purposes, it does not constitute a high-risk AI system. In that case, no high-risk requirements apply to the AI application, although transparency obligations may still apply (see section 1 on page 50).

An LLM application supporting public information provision will in most cases not be intended for deployment in a high-risk application. Where this is the case, the requirements for high-risk AI systems must be met.

The high-risk application domains are:

- Biometrics
- Critical infrastructure
- Education and vocational training
- Employment, workforce management and access to self-employment
- Access to and enjoyment of essential public benefits and services
- Assessment of creditworthiness
- Insurance
- Law enforcement
- Migration, asylum and border control management
- Administration of justice and democratic processes

Where the application is intended for deployment in one of these domains, it is important to verify whether one of the specific use cases applies and whether any exceptions are relevant. See Annex III and Article 6(3) of the AI Act.

Also verify whether there is foreseeable misuse, whereby a user deploys the application for a high-risk application, even though this is not the intended use.

Note: the AI Act's description of essential public services reads as follows:

"AI systems intended to be used by or on behalf of public authorities to assess whether natural persons are eligible for essential public benefits and services, including healthcare, or to grant, restrict, revoke, or recover such benefits and services."

If LLM applications deployed by public authorities are not used for any of these purposes, the application does not qualify as a high-risk application on this basis.

Nomenclature

Description of terms used in the validation framework. Terms on this list are marked in *italics* throughout the framework.

Term	Description
A/B testing	A/B testing is a method whereby two versions of the LLM application are compared in order to determine which per-forms better.
Benchmark dataset	A benchmark dataset is a collection of test scenarios, relevant to the specific context in which the LLM application is used, employed to evaluate and compare the performance of the application.
Chunking strategy	A chunking strategy is a technique whereby large amounts of texts are divided into smaller, more manageable segments for more efficient processing by large language models.
CI/CD	A methodology whereby software is tested automatically, so that software can be deployed more quickly and reliable.
Classification	The process whereby data is automatically assigned by an algorithm to predefined categories based on its characteris-tics.
Content filter	A content filter is software that filters the input (what the user shares with the LLM application) and the output (what the LLM responds) to detect and block harmful, unsafe, or undesirable content.
Fine-tuning	Fine-tuning is the process whereby an existing LLM is further trained on specific data for a specific LLM application in or-der to improve performance.
Front-end	A front-end is the part of the LLM application that users see and interact with.
Jailbreaking	Jailbreaking is the circumvention of the built-in safety measures and content filters of an LLM application in order to gen-erate content that would normally be blocked.
Ground truth label	The correct label, established by an expert, used as a reference to train or evaluate an algorithmic application.
Guardrails	Risk management measures that steer the behaviour of the LLM application to prevent undesirable or unsafe output.
Hallucination	An error where an AI system generates convincingly presented but incorrect or fabricated information.
Knowledge base	A knowledge base is the structured collection of information that the LLM can consult to provide accurate and context-specific answers.

Nomenclature

Term	Description
LLM-as-a-judge	LLM-as-a-judge is an approach whereby a large language model is used to assess the quality or correctness of the output of an LLM application.
Personal data	Personal data is all data related to a specific identifiable individual. This is a very broad category, encompassing indirect data such as the metadata of a user interaction (even when no name is processed). Whether a person can be identified depends on the combination of available data and the means available.
Performance metric	A numerical measure indicating how well an algorithmic system performs a task, such as accuracy, precision, and recall.
Prediction	A model-generated estimate of future or unknown outcome based on available data.
Production	The moment at which the LLM application is actually deployed and used in a real, live environment by end users.
Prompt -engineering -injections users- system-	Prompt engineering is the process of carefully designing and formulating prompts in order to have the LLM application generate the desired output as effectively and accurately as possible. Prompt injections are attacks in which malicious instructions are embedded in a prompt in order to mislead an LLM application or cause undesired behaviour. A user prompt is the instruction or question that a user gives to an LLM (application) to elicit a specific response or complete a specific task. This may also be a chat message or query. A system prompt is the input or instruction provided to an LLM in order to elicit a particular response, behaviour, or output.
RAG-application	A Retrieval Augmented Generation (RAG) application retrieves information from external sources to provide an LLM with context, enabling it to generate more accurate and relevant answers.
Red-teaming	The systematic testing of an LLM application by assuming the role of an attacker in order to identify vulnerabilities and weaknesses.
WCAG	The Web Content Accessibility Guidelines (WCAG) are international guidelines describing how to make websites and digital content accessible to everyone, including people with disabilities.

Development of the Validation Framework ‘Responsible deployment of Large Language Models (LLMs) for public information provision’

This validation framework was produced through collaboration between the parties listed below, as a result of the ‘Responsible AI in Practice’ call by the SIDN Fund and Topsector ICT.

Algorithm Audit

NGO Algorithm Audit is a European knowledge platform for responsible AI. The platform brings together data scientists, legal professionals and ethicists to formulate solutions to value-driven questions that are central to the deployment of algorithmic systems.

Deloitte.

Deloitte is a global advisory and accountancy firm offering services in the areas of audit, consulting, financial advisory, risk management, and taxation.

de Rechtspraak

The Dutch Judiciary is the collective of courts and judges that independently administer justice and resolve disputes in the Netherlands in accordance with the law.

T&T Data Consultancy

T&T Data Consultancy is a Dutch data and AI advisory firm that helps organisations set up smart data solutions, implement AI, and engage people in that transition.

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Technical University Eindhoven (TU/e) is a Dutch university focused on research and education in technology, engineering and innovation.

This project was made possible by:

SIDNfonds

TOPSECTOR ICT dutch digital

The validation framework ‘Responsible Use of Large Language Models (LLMs) for Public Information Provision’ was developed under the CC BY 4.0 licence. This means the framework may be freely used, shared, and adapted, including commercially, provided that the creator is credited in accordance with Creative Commons conditions. No rights may be derived from this framework.