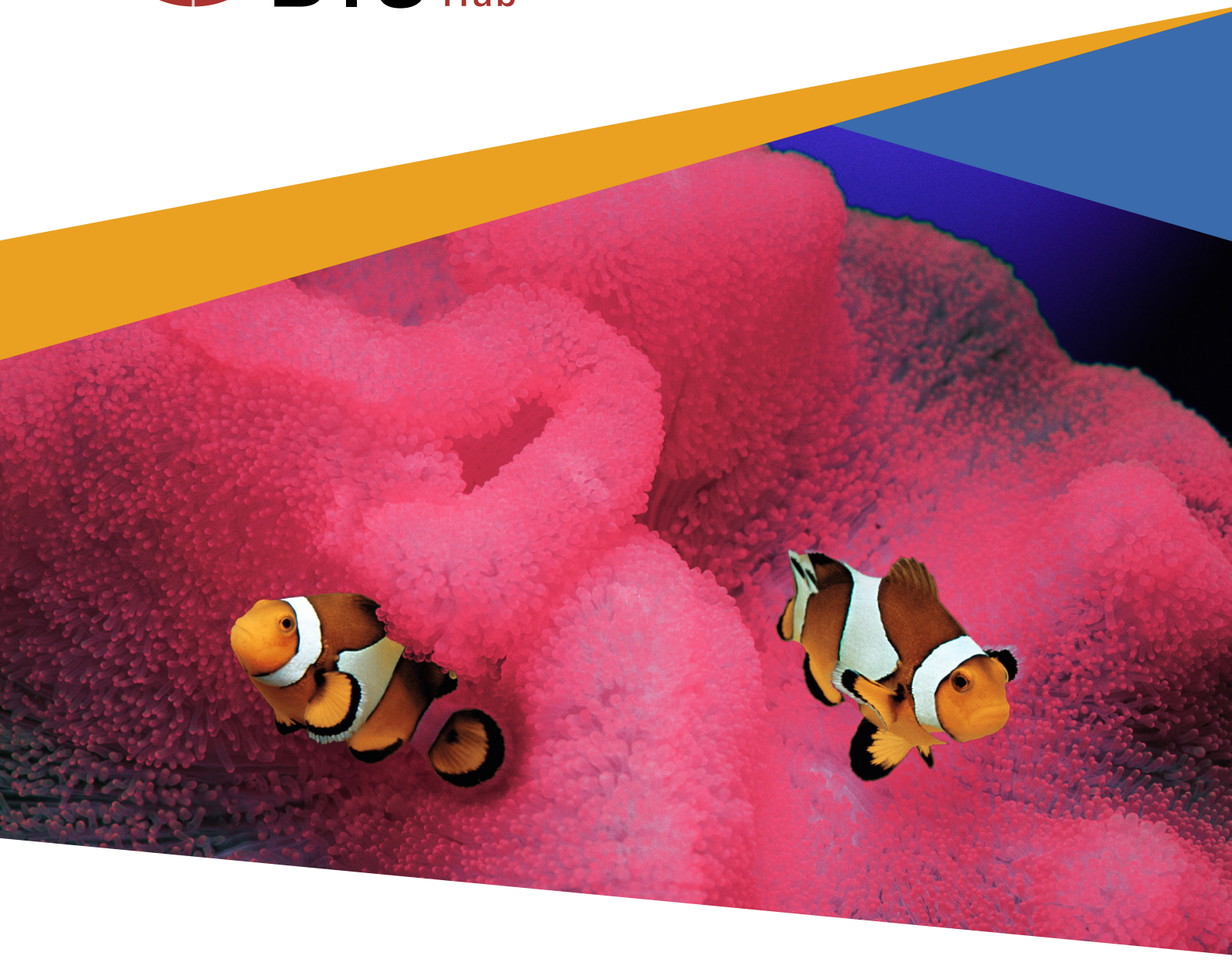
BIS Innovation Hub



Project Symbiosis

Part 2: Technical report

Exploring AI for scope 3 accounting and transition finance

October 2025

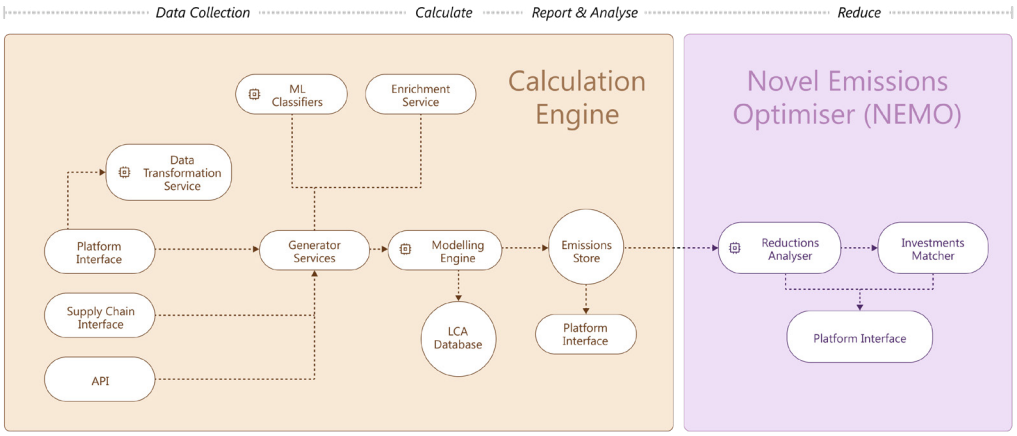
Contents

Overview

As set out in Part 1, the first goal of Project Symbiosis is to explore how advanced data techniques and artificial intelligence (AI) can be leveraged to more accurately collect, interpret and calculate scope 3 emissions and other impact data in corporate supply chains. For the purpose of this goal, the project performed **applied technology research** that explores and explains the AI techniques that could be used to achieve this. The second goal of Project Symbiosis is to explore approaches for identifying opportunities to reduce such scope 3 emissions. The third goal is to design a "matching engine" to match suppliers with funding sources to decarbonise the supply chain (referred to as financeable emissions reduction opportunities). For the purpose of these goals, the project developed a proof of concept (POC) referred to as the **Novel Emissions Optimiser or NEMO**.[1][2]

Graph 2.1

Architecture of the explored Project Symbiosis platform



Taken together **the applied technology research and NEMO** provide a blueprint that seeks to address **a number of the challenges,** with a specific focus on:

- Reducing the friction and effort involved in **data collection** to calculate a corporate carbon footprint (CCF). High levels of friction consistently result in lower levels of footprint accuracy and a reduced focus on decarbonisation measures.

- Increasing the accuracy of emissions **calculations** by utilising a more granular set of emissions factors and using modelling techniques for targeted proxy selection of missing data points consistent with a range of **reporting** regimes.

1 Refer to Part 1 for further context on the project and the choice of the name NEMO.

- Helping users identify and understand the **reduction potential** of impactful decarbonisation measures, with the ability to forecast and align projected emissions to a target.

I. Data collection

Data collection is among the most challenging aspects of carbon accounting due to fragmented supply chains and inconsistent data. Accordingly, in Project Symbiosis we explored solutions that simplify data collection, automate the transformation of disparate data formats, support supplier engagement and fill data gaps with logical assumptions. While emerging AI shows promise in further automating the process, its effectiveness to date has been poor, warranting further research and development.

Data collection covers the set of features used to aid users in the collection of the broad sets of data required to calculate a CCF. Given that the accuracy of the calculations is **directly related to the quality** of the data collected, reducing the **friction** of collecting **high-quality data material,** has been a focus when it comes to feature development.

Data collection is frequently stated by users to be the most painful part of the carbon accounting journey, for a few reasons:

- The breadth and variety of required data means engagement with a broad set of stakeholders, the vast majority of whom have limited knowledge about the data requirements for a carbon calculation.
- Data are often siloed, inconsistent and incomplete due to a lack of an overarching and established data strategy and the governance to support it.
- Access to data from the supply chain is very limited. The vast size and complex nature of supply chains makes individual engagement difficult if approached via traditional means.

To address these problems, features relating to a number of key requirements were explored. These features enable:

- Customers to upload data in any format, to reduce the friction associated with manual manipulation.
- The transformation of data in varying quality and formats into usable data for the calculation.
- The extension of the reach of users, powering data collection from their supply chain.

Explored features focused on product and logistics data sets due to their relative importance for supply chain emissions:

Table 2.1
Summary of key product and logistics data points

Data set	Data Examples
Product	• Product Details (Title, Category, Supplier) • Bills of Materials (BoM) (Material Composition, Product Mass) • Supply Chain Details (Energy Type, Energy Consumption)
Logistics	• Origin & Destination Details (Country, Postal Code) • Line Item Details (Quantity, Type) • Package Volume and Mass • Transport Mode

A. Data transformation

Effectively utilising customer data sets for the emissions calculations requires that the data are normalised and mapped to the data format required for the calculations. In a world in which all customer data followed a consistent format and all third-party systems exported data in the same way, transformation of the data would be a low-effort, one-time exercise. However, this is not the case. It would be unreasonable to assume that all customer data are of the same quality, although third-party systems exporting data consistently for the purposes of carbon accounting may become more routine in the future.

Normalisation – cleaning and structuring the data so that it follows a consistent format. This includes standardising units of measurement (eg converting weights from pounds to kilograms), resolving inconsistencies (eg different spellings of the same material) and ensuring that data fields are complete. Normalisation is essential because inconsistent or incomplete data can lead to errors in calculations.

Mapping – aligning the customer's data fields with the format required for emissions calculations. As customers may provide data in various formats – using different column names, categorisations or levels of detail – the data need to be translated into a standardised structure that matches the calculation framework. This ensures that the correct inputs are used in the emissions model, leading to reliable results.

Common examples of data quality and formatting issues across the product and logistics data sets can be seen in Table 2.2.

Table 2.2

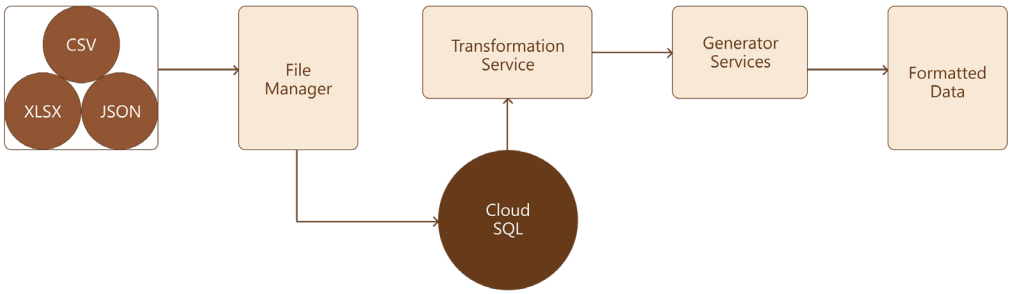
Summary of common product and logistics data quality issues.

Data set	Examples of common supply chain data issues
Product	• Bills of materials (BoM) formats and contents vary per supplier, resulting in many different formats. • Final product mass is often separate to the BoM and is incomplete or inconsistent. • Conflicting data between BoM and the product master data stored in another system.
Logistics	• Exports always vary between logistics providers, of which there are multiple per type of shipment eg inbound, e-commerce, wholesale. • Mixture of aggregated and individual shipment data, without a clear indication on the split. • Destination information is often separate to the logistics providers' exports.

To enable customers to upload these disparate formats, rather than requiring them to **manually populate templates or forms in a user interface,** a **data transformation pipeline** has been proposed **(Graph 2.2)**.

Graph 2.2

Overview of the explored data transformation pipeline



The **file manager** is an object-based, cloud storage solution which allows customers to upload data (eg a product bill of materials) in a number of supported formats; CSV, XLSX and JSON. The user is able to upload these files through the user interface, while **providing context** such as the GHG Protocol category it belongs to and a brief description of the data. In cases where the user has large files (> 500 MB) or wishes to upload a large volume of files, data can be uploaded through **file transfer using SFTP.** Once the files have been uploaded, the user's main responsibilities are completed, ie they do not have to perform any data transformation themselves.

The files are subsequently loaded into a cloud-based SQL database, where the data are available for analysis and subsequent transformation. The **transformation service** is an application which enables a data engineer to **efficiently write** and store script-based transformers to transform the data into the expected calculation format. The application has many functions, including **automated data validation checks** and the ability to **re-use custom functions** between transformers, that have optimised the time it takes for a data engineer to implement a transformer. Once complete, the transformed data are sent to one of many **generator services**, where the calculation processing begins.

AI in data transformation

Data transformation is a very manual task, why has it not been automated before?

Recent developments in AI have opened up genuine opportunities for automation but – due to the complexity of the specific data quality issues described in this report – the required levels of accuracy and performance have not yet been reached. Given the criticality of accurate input data for emissions calculations, the requirements for accuracy and performance have to be high.

Explored feature – an AI agent to automate file analysis and produce a data mapping that a data engineer can implement. The agent can trigger a set of tools (eg manually pre-defined function calls in Python) which enable it to analyse the data and propose the transformation mapping. Recognising the diversity of data that can exist within a file, the agent is particularly focused on analysing the data, rather than being limited to simple column matching, to identify the anomalies or edge cases that often drive the transformation effort.

Testing approach – comparison of automatically generated mappings to manually generated mappings, using the following key metrics:

- accuracy – percentage of correctly mapped customer fields to the target data schema; and
- completeness – percentage of mapped customer fields out of the total number of target data schema fields.

Qualitative performance – to date, the performance of this feature has been poor, which is a reflection of the complexity of the challenge due to the variety and context-specific nature of many customer exports. Although context is gathered during the data collection process, it currently appears to be insufficient to enable an agent to deduce and infer in the same way that a human can.

Performance is also impacted by the limited availability of labelled data sets, ie manually generated mappings, due to the niche nature of the use case compared with common use cases for which large labelled data sets are available publicly. This prevents fine-tuning from currently being a viable method of performance improvement.

Outlook – in the absence of applicable labelled data sets becoming available, performance improvements will be dependent on the performance improvements of the underlying LLM, changes to the core prompt structure (prompt engineering) and the use of retrieval-augmented generation (RAG) to enrich the prompt with previous mappings for additional context.

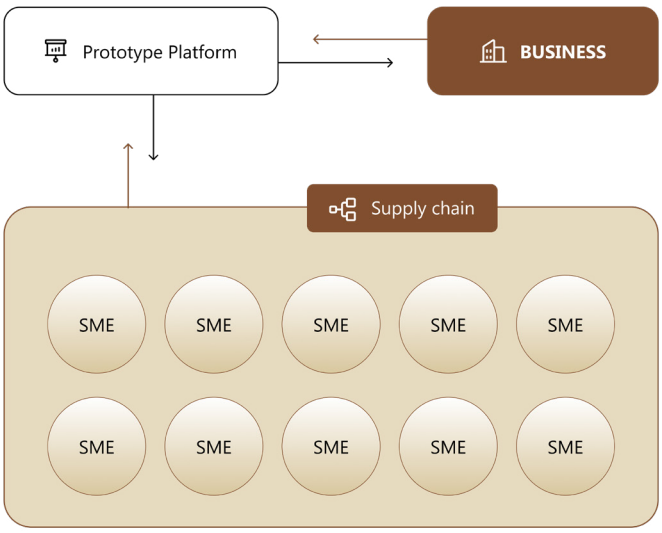
B. Supply chain data collection

Data transformation addresses the problem of users spending a lot of **time collecting and entering data manually** into templates and forms. However, it does not address issues related to the user **not having the data in the first place.**

To address this problem **a supply chain data collection approach** was explored through which SME users would be enabled to provide energy and resource consumption data for their facilities and the traceability of their own suppliers, enabling continuous construction of a supplier map. Ultimately, this would allow for a tier of suppliers to be mapped back to a product that the corporation user wishes to calculate the footprint for.

Graph 2.3

Data collection from supply chain partners



The developed solution could include many features to maximise the accuracy and effectiveness of the data collected, including:

- strong, typed validation (ie strict data type adherence requirements) to minimise data quality issues;
- custom data collection cadences with automated reminders and alerts;
- integration with email and other data formats to cater to a diverse supply chain;
- collection and parsing of transaction certificates; and
- dynamic language translation with the necessary context.

For cases in which the user already has access to supply chain primary data, the conceptual solution would enable uploading the supply chain data to the file manager, where it is transformed and available for use in the calculations. This user flow could also function in mixed scenarios where the user has partial supply chain primary data and wishes to use that to trigger ongoing data collection through the supply chain interface.

AI in supply chain data collection

Solution feature – using GPT-4o to parse and store data related to transaction certificates, which are documents that certify that the products in the transaction comply with a standard, eg the Global Organic Textile Standard. With this feature, once a supplier uploads the PDF through the supply chain interface, the PDF file is parsed and the extracted and structured data are sent to a target API. Both the user and supplier are then able to view the data in a structured manner through the interface, reducing the time spent on analysis and auditing.

Testing approach – comparison of automatically parsed certificates with manually parsed certificates, using the following key metrics:

- accuracy – the percentage of correctly parsed customer fields to the target data schema; and
- completeness – the percentage of parsed customer fields out of the total number of customer fields in the certificate.

Qualitative performance – although it sounds similar in nature to the previously mentioned data transformation AI agent, the performance here is much better due to the standardised and controlled nature of the transaction certificate format. An expected and consistent format, with limited variation in data, results in a level of accuracy and completeness that is sufficient for this to be used in a live production environment.

Although the current application is niche, it does highlight the potential for expansion to address the transformation of PDFs which express primary data in a consistent manner, eg purchase orders or utility bills.

Outlook – for the current application, the performance is only expected to marginally improve in line with the general performance improvements of the underlying large language models (LLMs). Expansion to other data sets contained within PDFs will likely require schema-driven extraction through adjustments to the prompt or the use of post-processing tools available to the LLM.

C. APIs and integrations

While data transformation is effective at reducing user effort when it comes to data collection, it best caters to batch processing rather than real-time collection. Although real-time collection and processing is currently a lower priority for users, as decarbonisation roadmaps are ramped up, the **need for a real-time feedback loop will increase.** A conceptual solution was explored that would rely on a number of **APIs and out-of-the-box integrations** to third-party solutions. This was in anticipation of a growing demand for this requirement and with the general reduction in effort enabled by these technologies in mind.

First, public representational state transfer application programming interfaces (REST APIs) could enable users to integrate any of their software applications which contain data sets related to products, logistics and facilities, and are therefore **not constrained** by the breadth of software applications in use by the corporate users. Once the user has integrated the API, a **real-time collection and calculation** pipeline is established, allowing the user to see the emissions impact of a product or activity **within minutes** of it happening. To ensure an appropriate level of data quality, APIs employ strict validation to prevent egregious data quality issues.

For the SME, participation would be incentivised through access to reduction analysis and transition finance as discussed later in Part 2, facilitating improved competitiveness in an increasingly carbon-regulated environment.

A REST API is a way for different software systems to communicate over the internet. It allows applications to send and receive data in a structured way eg enabling a corporation to send product data from an enterprise resource planning (ERP) system to the Project Symbiosis platform in an automated way.

Second, collection could further be enabled through incorporation of out of the box integrations which address user problems similar to those addressed by APIs, although without the flexibility. Integrations such as these exist for common technologies used within supply chains, including data warehouses, e-commerce platforms and ERP platforms.

While integrations can be used by any type of user, they are most commonly used by smaller corporations who have limited resources (to collect data) and less diverse data sets, increasing the percentage of data contained within one of these systems.

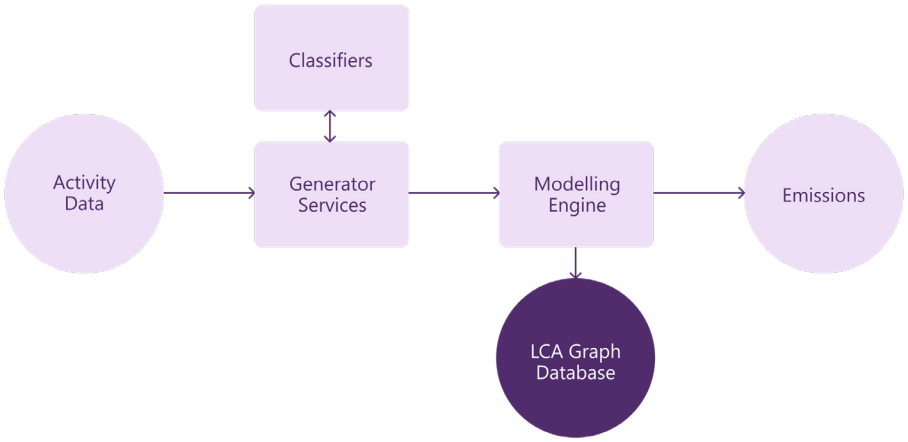
II. Impact calculation

Project Symbiosis explored solutions for turning real-world data into reliable emissions estimates, even when inputs are incomplete or inconsistent. Central to this is the explored modelling engine approach, which uses a graph database and probabilistic inference to identify contextual relationships, intelligently fill data gaps and maintain calculation accuracy across diverse scenarios, even with incomplete inputs.

This section covers the exploration of features which are responsible for using the collected data to calculate the **environmental impact** of the user's products and business activities. While the explored data collection features aim to help the user to maximise the quality of material data that can be collected, the reality is that data points will always be missing. Accordingly, a calculation pipeline which can **flexibly cater** to a **mixture of data quality** is necessary.

Graph 2.4

The proposed calculation pipeline



The key components of the explored calculation pipeline approach are:

- **Generator services** – software services responsible for some of the calculation business logic, integrations with supporting applications such as classifiers or third-party services and generating the modelling engine queries.

- **Classifiers** – AI-based classifiers, explored in the **AI for data classification** section below.

- **Modelling engine** – software engine utilising a graph database responsible for emissions calculations, using limited to extensive data inputs.

- **Life cycle assessment (LCA) database** – an extension of the graph database which contains the LCA data points, including emissions factors.

It is within these key components that the methodologies dictated by existing standards have been embedded. This is to ensure that they are executed in a consistent and compliant way.

Quick reference glossary

- **Life cycle assessment** (LCA) – a methodology for assessing the environmental impacts associated with a product's life cycle from raw material extraction to end of life.

- **Probability distribution** – a mathematical function that describes the likelihood of different outcomes in an experiment.

- **Prompt engineering** – the practice of crafting instructional prompts to guide the behaviour and output of large language models for specific tasks and high-quality outputs.

- **Zero-shot learning** – a machine learning approach in which a model makes predictions about tasks it has never seen before by leveraging general knowledge or contextual understanding acquired during training.

- **Routing logic** – a mechanism that dynamically selects which expert (in a mixture of experts (MoE) model) to activate for each input.

- **JavaScript Object Notation** (JSON) – a format for storing, structuring and transmitting data between systems.

- **Reinforcement learning from human feedback** (RLHF) – training technique in which a model learns desired behaviour by receiving reward signals based on human feedback.

A. Generator services

Generator services play a critical role in the explored emissions calculation process by handling key aspects of **calculation logic,** specifically deriving and standardising data points before they reach the modelling engine. This **split-responsibility approach** ensures **both flexibility and accuracy,** adapting to the different types of data that users can provide while maintaining the integrity of emissions calculations.

A key benefit of this approach is that it would allow users to submit **simpler, more commonly available data points,** which the generator service can use to derive necessary details for the calculation. For example, instead of requiring users to input a precise transportation distance (which may not always be known), the service can calculate it automatically from an origin and destination, ensuring that emissions estimates remain reliable even when complete data are not provided.

Beyond filling in missing data, the generator service can also **normalise and validate inputs** to prevent errors. Customer data sets often include inconsistencies such as:

- Different units of measurement (eg distances reported in miles vs kilometres).
- Variations in naming conventions (eg “road freight” vs “truck transport”).
- Gaps in data that require enrichment from reference data sets.

By standardising and structuring these inputs, the generator service ensures that the **modelling engine receives clean, accurate data,** making the overall calculation process more efficient and scalable. **Table 2.3** provides an overview of the generator services explored in Project Symbiosis.

Table 2.3
Generator services explored

Generator Service	Description	Evaluated in Project Symbiosis
Product	Service for handling product-related and supply chain data used for calculating a product carbon footprint (PCF).	• Standardising categories and materials. • Enriching individual products with facility-level supply chain data. • Normalising product and component weights.
Logistics	Service for handling logistics data used for calculating emissions across a logistics network.	• Calculating distance based on origin and destination. • Selection of transport mode based on route. • Selection of shipment legs based on shipment type.
Facilities	Service for handling facility-level energy consumption data, used for calculating scopes 1 and 2.	• Extrapolation of energy consumption when consumption is incomplete for a given period. • Estimating energy consumption based on floor area.
Business Operations	Service for handling data points related to scope 3 categories such as business travel and employee commute.	• Calculating distance based on flight departure and arrivals. • Calculating working from home time based on commuting frequency.

Why is a generator service required? Could the modelling engine be fed the input data directly?

While it might seem simpler to send raw data straight to the modelling engine, this would lead to major issues. Suppliers provide data in different formats, with inconsistencies in units, naming conventions and missing details. The generator service ensures all data are structured correctly and that data gaps are filled, making the modelling engine's job more efficient and accurate.

There are no strict rules for whether calculation logic should be handled by the **generator service** or the **modelling engine.** Rather, this decision is driven by the **complexity of the calculation** and the **number of assumptions required.**

- **Simple calculations with fewer variables** (eg deriving missing activity data points like transportation distance from an origin and destination) are typically handled by the **generator service.** This ensures that the **modelling engine receives structured inputs,** keeping the calculation process efficient.

- **More complex calculation steps with multiple interacting variables,** such as those in a **product carbon footprint (PCF) assessment,** are performed in the **modelling engine.** In these cases, applying **simplistic assumptions too early in the process** could introduce greater uncertainty. By keeping such calculations in the **modelling engine,** we maintain the flexibility to use more advanced statistical models, probabilistic distributions or additional data sets to improve accuracy.

This approach **balances flexibility and accuracy,** ensuring that simpler data transformations happen early in the pipeline while **more detailed, assumption-sensitive calculations remain within the modelling engine** for greater precision.

AI for data classification

Solution feature – the processing of thousands of products daily necessitates a model capable of the automated classification of product details within a set of taxonomies to the required level of accuracy.

This classification model uses a hierarchical framework that reflects the solution's hierarchical taxonomy, allowing for classification at different levels of the taxonomy based on the quality of the product information received.

OpenAI ada-002 was explored for embedding textual data – an industry standard approach for turning text into numbers so that a computer can understand and work with it. This feeds into a classification architecture composed of multi-class support vector machines (SVMs) selected for their overall performance.

The process starts with a broad categorisation of the product, progressively narrowing down through the taxonomy tree based on the predictive confidence at each taxonomy level, until the confidence level no longer meets a set threshold.

This approach ensures that a minimum level of classification based on a taxonomy can be achieved with limited data (eg only a product title), while offering the ability to move towards a more granular classification when more context is available (eg product category, description, attributes).

The approach also addresses one of the key challenges with classification based on a taxonomy, which is that due to the vast breadth of product categories sold by consumer goods retailers, it is not practical to have a taxonomy class for every possible category. Hierarchical classification allows for products to be classified based on association with similar products, even if the explicit category is not available, eg a "padel ball" being classified as a "tennis ball", or a "smock" being classified as a "top". As the association is based on the functional attributes of the product, there is minimal impact when it comes to the emissions calculation.

Testing approach – comparison of automatically classified products against a manually labelled data set of classified products using the following key metrics:

- Accuracy – percentage of classifications that are an exact match with the labelled classification.
- Loss – an aggregate score across all classifications, accounting for those that are an exact match, under-classified (too coarse), over-classified (too granular) or misclassified when compared with the labelled classification. The lower the loss, the higher the performance. The use of a loss score allows for a holistic assessment of the model's performance, making it a more useful metric than accuracy for this particular use case.
- Hit rate – percentage of products which have been classified by the model.

Qualitative performance – a motivation to switch to a hierarchical classification model was the performance of the previous model, which could only classify to a taxonomy level if the model had been trained to do so. In addition to being wholly dependent on the training approach, it had a tendency to classify based on keyword matching, ignoring other key words, eg "ratchet set" or "spice set" being classified as "duvet set". The switch to the hierarchical model significantly improved the accuracy of the model in this regard, particularly minimising the number of misclassifications at the top of the taxonomy tree.

The hierarchical classification model led to an increase in the hit rate and a reduction in the loss, the latter was particularly driven by the model's ability to classify products one level away from the true classification, rather than being unable to classify the product at all. The use of OpenAI ada-002 means that the classification is language independent, negating the need for a pre-processing translation step, which often results in literal translations and therefore a loss of context. Targeted testing by comparisons to a non-English labelled data set showed performance improvements across all metrics.

Despite the clear improvements, the model performance still showed sensitivities to input data which deviated from the data that has been used for training. Specifically, deviation resulted in lower performance across all metrics. While this is expected behaviour, the extent to which performance suffered has been a surprise and presents a challenge for a classification pipeline which is intended to be customer agnostic.

Outlook – given the sensitivity of performance to variations in input data, future efforts can be focused on addressing this through the use of increasingly curated training sets and better alignment between the balance of the source of training data and production data. While data sets are available for this use case, to avoid issues of homogeneity in the data, these data sets will be supplemented with synthetic data sets (ie generated by an LLM) based on examples of existing production data.

B. Modelling engine

As has been highlighted, emissions calculations often rely on **incomplete data sets** as companies may not have precise carbon footprint data for every material, process or location. Instead of making rigid assumptions or using fixed default values, **statistical and probabilistic modelling techniques** can be used to infer missing values with high confidence, improving **the accuracy of emissions calculations.**

To explore this, we evaluated a series of modelling techniques (the **"modelling engine"**) designed to maximise the accuracy of emissions calculations, even when only **limited primary data** are available. The **modelling engine is built on a graph database (Neo4j)**, which allows it to represent **complex relationships between products, materials and processes** more effectively.

To achieve this, the modelling engine is structured around **two planes of data**:

- **Data plane** represents **real-world** products, processes and materials, along with their emissions impact. This is where **LCA data** are stored, capturing emissions at various life cycle stages.
- **Ontological plane** acts as the **organising layer** that structures the data plane. It defines **relationships, classifications and regional variations,** allowing for **harmonisation across diverse data sets.**

A **plane** represents a distinct layer of information in the model. Each plane serves a specific role, ensuring that real-world emissions data are both structured and interpreted correctly.

Neo4j graph database is a graph database, which means it stores data as connected relationships instead of rigid tables like traditional databases. Instead of rows and columns, it organises information as nodes (data points) and relationships (connections between them). This structure makes it ideal for modelling complex networks, such as supply chains or product life cycles, where different elements are highly interconnected.

Why does the modelling engine use two planes?

Think of the modelling engine like a library, but instead of books, it organises emissions data.

- The **data plane** is like the **bookshelves**, where all the real-world information (eg LCA models, emissions data, product footprints) is stored.
- The **ontological plane** is like the **library catalogue,** which helps organise and classify everything so that users can find what they need, whether they are searching by author, subject or topic.

Without the **ontological plane,** it would be like having a massive library **with no catalogue.** Books would be piled up randomly making it nearly impossible to compare information or find the right data efficiently.

By separating the **data (books)** from the **structure that organises them (catalogue)**, this approach ensures:

- **Separation of layers** – just as books and a library catalogue serve different purposes, real-world emissions data are kept distinct from **abstract definitions** like taxonomies and classifications. This prevents the emissions model from being **rigid or limited** to a single classification system.
- **Clarity and consistency** – every piece of data are **explicitly defined and categorised,** reducing misinterpretations, just like a library catalogue ensures books are not misplaced or mislabelled.
- **Scalability and integration** – the **ontological plane acts as a universal translator,** allowing different data sets (even if they use different categories or units) to be **merged and compared seamlessly.** This is similar to how a library catalogue enables books from different genres, languages or publication years to be found and used together.

The **generator service** interacts with the **modelling engine** through a **query language** that aligns with the **LCA models in the data plane**. This query language is designed to be **highly flexible**, allowing it to:

- **Enable detailed context** about the product/activity to be passed to the engine that is used to **identify relationships** between the product and comparable data in the database.
- **Provide primary data** when available and make it clear when the data are not available so that these **data gaps** can be filled.

Relationship – in a graph database, a relationship is a structured connection between two data points.

For example, a cotton T-shirt might have relationships with the:

- materials it is made from (cotton fibre, dyes);
- processes used in its production (knitting, dyeing, assembly); and
- regions where it is produced (China, India, European Union).

These relationships help the modelling engine to structure its search for relevant data, ensuring that missing values can be inferred based on contextually similar data points.

While relationships help narrow down potential substitutes, the statistical modelling process determines the most appropriate proxy to use in emissions calculations.

For example, if the emissions factor for a specific production method is missing, the graph database first identifies similar processes using structured relationships. The statistical modelling process then aggregates, compares and infers the most accurate estimate using probability distributions and quantile reconstruction.

This statistical modelling process follows four key steps:

1. **Probability distributions** – each potential substitute for the missing data point is **converted into a probability distribution**, reflecting its **likelihood as a valid replacement**. This ensures that instead of a single best guess, the system considers **a range of possibilities.**
2. **Cumulative density functions (CDFs)** – to compare different probability distributions, each one is converted into a **CDF**. This transformation standardises the data, allowing the system to compare **how likely** each potential substitute value is relative to others.
3. **Combining CDFs** – rather than selecting a single substitute, the **modelling engine aggregates multiple data points** to generate a **unified distribution**. The range of the random variable (x_min to x_max) is divided into 100 steps (s), and the weighted mean of all CDF values is calculated at each step.

4. **Quantile function reconstruction** – a **CDF alone is not actionable** for calculations, so it must be **converted back into a quantile function** using a two-phase approach including (i) percentile approximation and (ii) percentile interpolation. This ensures smooth and complete reconstruction of the quantile function.

Simplified

Think of filling in missing emissions data like solving a jigsaw puzzle, but with some pieces missing. Instead of guessing randomly, we use logical techniques to fill in the gaps as accurately as possible.

Probability distributions → sorting through possible puzzle pieces
What is happening?

- Instead of assuming a single correct answer, the system considers multiple possibilities for the missing data.
- It treats each potential value like a **puzzle piece that might fit**.

Imagine you are missing a piece of sky in your puzzle. Instead of grabbing any random piece, you **look at several blue pieces and rank them by how well they might fit** based on their shape and shade.

CDFs → comparing different puzzle pieces
What is happening?

- The system compares all potential values in a standardised way, making it easier to find the most likely substitute.

You take all the possible blue puzzle pieces and **lay them out in order,** from the **lightest blue to the darkest blue,** to see which ones are the closest match.

Combining CDFs → creating the best composite piece.
What is happening?

- Instead of picking just one substitute, the system blends.

Imagine **none of the blue puzzle pieces are an exact match,** but two or three are **very close.** Instead of choosing just one, you **cut and reshape** a piece by **combining the best parts** of each to create a nearly perfect fit.

Quantile function reconstruction → finalising the fit
What is happening?

- The system smooths out inconsistencies and ensures the final estimate is continuous and accurate.

After reshaping the puzzle piece, you **smooth out the edges,** making sure it fits seamlessly into the puzzle, so that it looks natural.

The end result is the use of data points in the calculation which, although not primary in nature, are expected to be significantly closer to the real-world value than a simple proxy or average value selected by a human alone. The combination of **planes, relationships and statistical modelling** allows the modelling engine to:

- **Accurately process incomplete data** by intelligently filling gaps.
- **Ensure transparency** in emissions calculations, as all assumptions and inferred values are based on structured relationships.
- **Scale across different industries** without requiring a rigid structure.

1. LCA database

The LCA database refers to the LCA data points which underpin the LCA models in the modelling engine. While not technically separate, conceptually, the LCA database refers to the collection of data points which are available for use in the calculations.

Life cycle assessment (LCA) is a method used to **measure the environmental impact** of a product, process or service **across its entire life cycle,** from raw material extraction to disposal or recycling.

LCA breaks down the full life cycle into key stages, such as:

- raw material extraction (eg mining cotton or metal ores);
- manufacturing and processing;
- transportation and distribution;
- use phase (eg electricity needed to operate a device); and
- end of life (eg landfill, recycling).

Each stage is assessed for its impact on **climate change, water use, energy consumption** and more, depending on the scope of the assessment. By understanding these impacts, businesses can make more informed decisions, reduce emissions and support sustainability goals.

To support accurate modelling, the database contains:

- **Primary and secondary data** about real-world activities and products.
- **Emissions factors** used to quantify the impact of materials, energy, transport etc.
- **Relational metadata** that links these data to taxonomies, geographies and life cycle stages.

These data points are sourced from recognised, reputable sources and the LCA database is actively maintained by **LCA experts** who ensure that the data:

- Reflects **current technologies and market conditions.**
- Accounts for **geographic and temporal differences.**
- Is suitable for both **detailed product-level footprints** and **higher-level estimations.**

This **continuous curation** ensures that emissions calculations stay up to date and relevant, especially as industries shift to more sustainable practices.

2. Impact category flexibility

Although the focus of the project and this report is on climate change, the modelling engine is able to calculate broader environmental impacts, specifically the 16 impact categories outlined in the product environmental footprint (PEF) method.[13]

Table 2.4
Overview of PEF's 16 impact categories

Impact Category	Unit
Climate change	kg CO_2 equivalents
Ozone depletion	kg CFC-11 equivalents
Human toxicity, cancer	CTUh (comparative toxic units for humans)
Human toxicity, non-cancer	CTUh
Particulate matter	Disease incidence
Ionising radiation, human health	kBq U-235 equivalents

2 European Commission, Commission Recommendation (EU) 2021/2279, December 2021, Annex I, eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A02021H2279-20211230.

Impact Category	Unit
Photochemical ozone formation, human health	kg NMVOC equivalent
Acidification	mol H^+ equivalents
Eutrophication, terrestrial	mol N equivalents
Eutrophication, aquatic, freshwater	kg P equivalents
Eutrophication, aquatic, marine	kg N equivalents
Ecotoxicity, freshwater	CTUe (comparative toxic units for ecosystems)
Water use	m^3 world eq deprived (deprivation-weighted)
Land use	Pt (soil quality index)
Resource use, fossils	MJ (megajoules)
Resource use, minerals and metals	kg Sb equivalents

All impact categories are calculated following the methodology outlined previously, utilising the emissions and emittable factors available in the LCA database. This ensures a consistency of approach.

III. Reporting and analysis

Explored solutions for results generation aim to deliver structured emissions data enabling detailed exploration, trend analysis and formatting that meet diverse regulatory and standards-based reporting expectations.

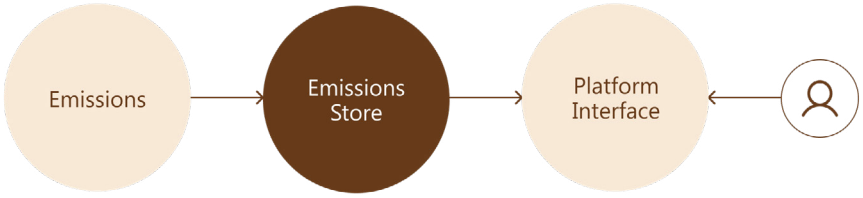
While data collection and calculation are the "how", **report and analyse** are some of the main drivers for the "**why**". Enabling users to report and analyse their footprints is a key requirement, particularly due to the **disclosure-related regulation** coming into force in **many markets** across the globe.

Alignment with regulations and standards is primarily achieved by the methodology and emissions factors used in the calculation, however there are a number of

requirements which dictate how results must be communicated to the user. These are elaborated upon in **Table 2.5.**

Graph 2.5

Report and analyse pipeline explored in Project Symbiosis



The key components explored for report and analyse are:

- **Emissions store** – a BigQuery database storing hundreds of millions of individual emitting entities and aggregated data sets.
- **Platform interface** – a React-based web application, in which users are able to view, analyse and export emissions results.

The emissions store is designed to cater for the wide range of emissions-generating activities which fall within the scope of a corporate or product footprint. The store schema is effectively split into two types of data:

- **Common metadata** – time, emissions, activity name, GHG Protocol category.
- **Activity-specific dimensions** – product details, logistics details, facilities details.

The exact data fields available reflect both the primary data provided by the user and facilitate the calculation of metrics that are required by the regulations and standards. A summary of the key metrics is provided in **Table 2.5.**

Table 2.5

Key metrics required for regulations and standards

Key metric	Examples of applicable regulations and standards
GHG emissions in kg CO_2e, by scope and GHG Protocol category	GHG Protocol corporate standard ISSB standards TCFD EU CSRD EU SFDR EBA ESG Pillar 3 SBTi

Key metric	Examples of applicable regulations and standards
Primary data used (percentage)	GHG Protocol corporate standard ISSB standards EU CSRD SBTi
Data extrapolated (percentage)	SBTi

The conceptual solution could therefore provide the user with the means to explore and export the emissions in a curated and structured way according to groups or "modules" of activities:

- **Company** – inventory of corporate emissions across scopes 1 to 3 for each CCF year, consistent with GHG Protocol requirements, with trends over time.
- **Product** – aggregated and per product emissions, broken down by product life cycle stage and scaled by the quantities of products purchased and sold.
- **Deliveries** – aggregated emissions for all deliveries across the logistics networks, broken down by delivery type, with absolute and intensity-related metrics.
- **Facilities** – aggregated and per facility emissions for facilities across scopes 1 to 3, broken down by location and energy type.

IV. Reductions

A centrepiece of Project Symbiosis is NEMO, the proof of concept software application developed to evaluate the ability of AI to identify financeable emission reduction opportunities based on varying emissions data. Two alternative pathways were explored, generator and matcher, which identified reduction opportunities with increasing precision over the evaluation period, in some cases approaching the competence level of a trained LCA professional.

A. Overview

Reductions covers the features that support users in identifying actionable reduction measures to lower the environmental impact of their organisation. Once emissions have been calculated, the challenge is knowing how to meaningfully reduce them:

- **Lack of internal expertise:** most organisations lack experts, with the necessary technical and domain knowledge to confidently identify reduction measures which are guaranteed to have a positive impact.

- **Complex supply chains:** decision-makers may have limited visibility or control over upstream activities (eg raw material production), which makes it cost-prohibitive to even assess the potential impact and feasibility of a measure.

- **Volume of possibilities:** there can be hundreds of potential measures across an organisation, so knowing where to start is overwhelming without expert-led prioritisation.

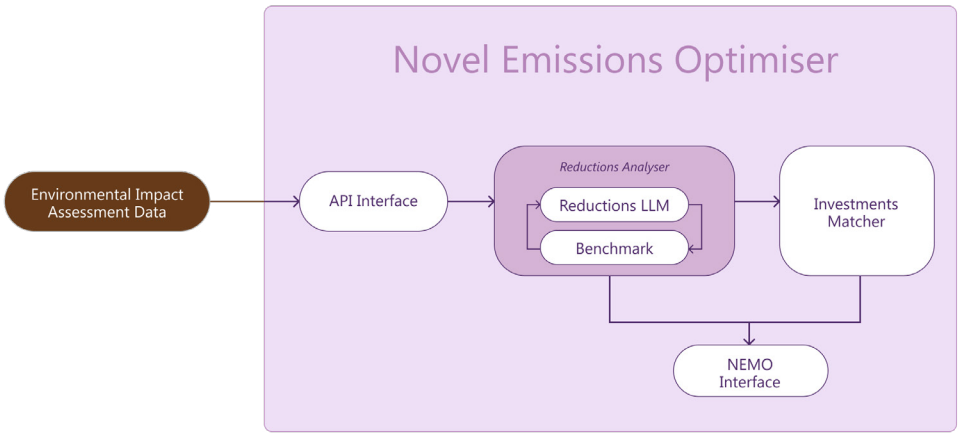
To help users navigate these challenges we developed NEMO as part of Project Symbiosis.

B. Novel emissions optimizer (NEMO)

NEMO is a proof of concept software application designed to identify and estimate the impact of decarbonisation measures when given environmental impact data as an input. It is intended to experiment with and showcase how an interface of this nature, combined with the use of LLMs, can help stakeholders bridge a knowledge gap in developing a decarbonisation roadmap. It also offers visibility on potential financing opportunities that may be necessary to bring the roadmap to reality.

Graph 2.6

Architecture diagram of NEMO



NEMO can be used with the output from any environmental impact assessment solution including a software-as-a-service (SaaS) platform, a consultant's report and an LCA report. Its standalone nature ensures broad compatibility with the breadth of environmental accounting solutions that are currently in use. This maximises the number of users who can benefit from the identified reduction measures.

1. Features and functionality

Table 2.6
Key metrics required for regulations and standards

Feature	Description
NEMO interface	The NEMO interface is the user interface of NEMO, where users will be able to upload environmental impact data, view their identified reduction measures and explore potential financing opportunities.
NEMO API	A REST API conforming to a defined schema that allows users to programmatically send environmental impact data to NEMO from carbon accounting solutions, databases or other third-party technologies. Given the potential volume of impact data, it is important to enable users to send data in an automated fashion and not limit them to manual upload through the user interface.
Reductions analyser – reductions LLM	The reductions LLM is a large language model (LLM) within the reductions analyser and is responsible for the identification of reduction measures when given product details as an input. The analyser supports two processing paths, which are detailed in this section.
Reductions analyser – benchmark	As one of the core features of NEMO, the reductions analyser will identify and assess the impact of reduction measures, using environmental impact data as an input. The benchmark feature is responsible for assessing the impact of the identified reduction measures, enabling prioritisation and aiding decision-making on the measures that will contribute most significantly to a reduction target.

Feature	Description
Investments matcher	The investments matcher is responsible for matching reduction measures to potential financing opportunities, eg bank loans.

2. Reductions analyser

The reduction analyser is the core feature of NEMO. It is designed to evaluate and optimise emissions reduction measures by leveraging AI capabilities and advanced computational techniques. Its design incorporates two key components:

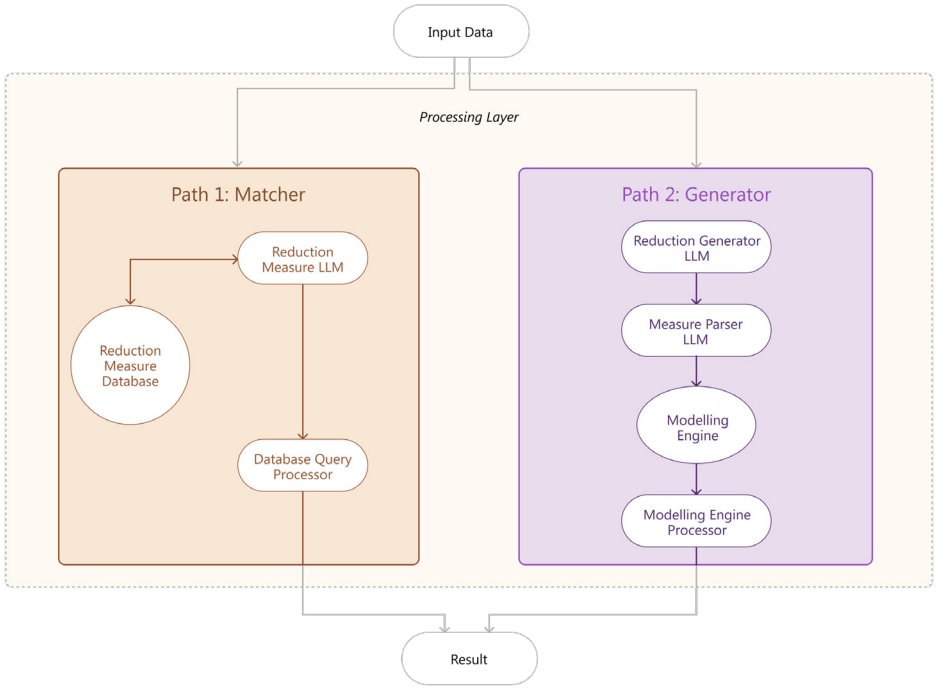
- **Reductions LLM** – the LLMs used to identify decarbonisation measures based on input data.
- **Reductions benchmark** – the logic used to calculate the impact of the decarbonisation measures in the context of a corporate footprint.

The analyser supports two processing paths:

- **Generator** – a processing path in which the reductions analyser is able to dynamically identify and evaluate reduction measures based on the inputs it receives, through integration with a third-party modelling engine. This could be thought of as a "bottom-up" approach.
- **Matcher** – a processing path in which the reductions analyser can only select reductions from a curated, pre-computed database of reduction measures validated by domain experts. This could be thought of as a "top-down" approach.

Graph 2.7

Overview of the processing paths in NEMO



The design decision to have two processing paths was driven by two challenges.

The first challenge is specific to the carbon accounting domain and is a result of the methodological flexibilities that are touched on in Part 1. There is a high likelihood that there would be a difference between the methodology and emissions factors of the baseline footprint provided as an input, and the methodology and emissions factors used to calculate the reductions. Therefore, it is necessary to have a processing path that can recalculate the baseline with the same methodology and emissions factors, enabling a like-for-like comparison with the identified reductions. The recalculation of the baseline and calculation of the reduction measure impacts is achieved with a third-party modelling engine that the user has access to and is not conducted by an LLM. This need is addressed by the generator in Table 2.6. An example of this problem is described below.

While the generator provides a higher level of accuracy, the NEMO POC was developed such that it conceptually could operate as a standalone platform, separate from the rest of the Project Symbiosis solution, including the modelling engine. To ensure that the friction of having access to a modelling engine did not limit the NEMO user base, an alternative processing path is available instead of using a pre-computed database of reduction measures curated by a team of LCA experts. This need is addressed by the matcher.

The combination of the two paths provides a standalone platform with capabilities that can cater to a range of users and needs, while providing opportunities to stress test the potential of AI to address the reductions use case.

Table 2.7
Benefits and limitations of the NEMO processing paths

	Benefits	Limitations
Generator	• Addresses methodological variabilities. • Opportunity to leverage continually developing LLM knowledge.	• Modelling engine requirement can limit user adoption. • Increased risk of less reliable outputs due to LLM hallucinations.
Matcher	• Reliable outputs due to the use of an expert curated reductions database. • Low barriers to user adoption.	• Reduction measures do not evolve without manual updates to the static database. • Reduction measure impacts are pre-computed, limiting the accuracy of the calculated impact.

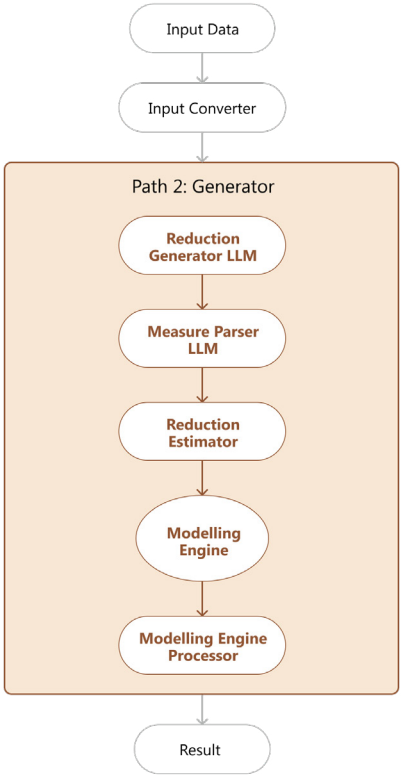
1. Generator

The generator path architecture consists of the following components:

- **Input converter** – the system takes the product information that users provide, such as materials, manufacturing details and energy usage, and converts it into a generalised format that ensures the LLM can easily identify key details while allowing for future extensibility in terms of the data that can be received.
- **Reduction generator** – an LLM used to identify reduction measures based on product and supply chain data. It interprets details such as material composition, manufacturing locations and energy sources, and uses its inherent knowledge to identify proven and appropriate measures.
- **Measure parser** – a separate LLM converts the generated, unstructured reduction measures into a structured data format.
- **Reduction estimator** – responsible for interfacing with the modelling engine and generating two queries: one that represents the baseline environmental footprint of the product and another that incorporates the identified reduction measures.
- **Modelling engine** – a third-party component which receives the queries and calculates the baseline and reductions footprints. This component is not technically part of the NEMO codebase but is interfaced with through an API.
- **Modelling engine processor** – responsible for using the baseline and reductions footprints to calculate a relative difference impact, which becomes part of the final output displayed to the user.

Graph 2.8

Overview of the generator processing path



The **reduction generator** uses an LLM that is not constrained by a list of curated reduction measures. Accordingly, the suitability of this kind of AI for helping stakeholders identify reduction measures can be properly assessed – a key objective of Project Symbiosis. Using the LLM's inherent knowledge to identify reduction measures does come with a risk of hallucinations, but the potential upsides are:

- Measures will stay current with the emergence of new materials.
- Measures will evolve as technologies and processes develop.
- Manual intervention by LCA experts to update a curated reduction measures database is not required.

Hallucinations – in the context of artificial intelligence, hallucinations refer to instances when an AI system, especially an LLM, generates information that is factually incorrect, misleading or entirely made up, despite sounding plausible. These errors can occur because the model predicts words based on patterns in data and not on a verified understanding of truth.

Given the risk of hallucinations, which could result in reduction measures that are inaccurate, inactionable or generally misleading, the generator path includes a number of mitigations:

- Testing of different models and configurations to identify the most accurate setup, specifically comparisons to a manually labelled data set.
- Validation of numerical data points, within the measure parser, through expected ranges.
- Modelling engine applicability, specifically that for a measure to be calculated, the engine needs to have support for the underlying details of the measure. For example, it would not be possible for the engine to calculate that a blanket is made of water, because water is not a supported fibre within the engine.

The selection criteria for identifying the LLM to use in the reduction generator were:

- Ability to generate accurate, realistic and granular reduction measures from supply chain and product data.
- Production grade API availability.

After testing and evaluation, the LLM selected for use was OpenAI o3. Details on the performance of the models across various tests are discussed in the following paragraphs.

To maximise the quality of the output, a core prompt structure was designed, the culmination of multiple iterations to identify the prompt that yielded the best performance. The design of this prompt reflects the need to guide the LLM's generative capabilities without excessive rigidity. By framing the prompt as a practical task ("I'm trying to..."), it simulates a real-world advisory use case. The constraints around functional and aesthetic properties reflect the need to balance intended product use with environmental impact. The prompt also anticipates the potential for hallucination by asking for explicit callouts when assumptions are made, giving necessary context for the evaluation of the model's outputs.

"I'm trying to find the best way to reduce the climate impact of my product and its supply chain. Suggest up to five different measures one could potentially take based on the product information below. Prioritise the measures by their estimated potential impact, where the highest comes first. We're mainly trying to optimise for climate change impact and not other impact categories. Measures should not significantly modify the functional and aesthetic properties of the product. Make assumptions about commonly used processes and highlight when assumptions were made to increase the specificity of the measures.

List each measure as a numbered item with a line break between items, like:

1. [First measure]

2. [Second measure]

Product information:

{{supply_chain_data}}"

The prompt consists of key elements that are highlighted in Table 2.8.

Table 2.8
Key elements of the generator prompt

Prompt element	Purpose
"Suggest up to five different measures..."	Indicates that a selection of reduction measures are desired, but limits output scope and promotes concise, high-impact suggestions.
"Prioritise by estimated potential impact..."	Encourages implicit ranking logic, aligning outputs with impactfulness.
"Optimise for climate change impact..."	Narrows scope to the relevant impact category (GHG), filtering out unrelated advice.
"Do not significantly modify the functional or aesthetic properties..."	Reflects real-world constraints in product design and brand fidelity.
"Highlight when assumptions were made..."	Promotes transparency and specificity, useful for downstream validation.
Numbered list format instruction	Standardises output format to ease evaluation and comparison.

A key design principle with the generator path was to provide a mechanism to mitigate the variances in methodology and emissions factors that exist. The use of a modelling engine means that a baseline footprint can be recalculated following the same methodology as the reduction measure calculations, enabling a like-for-like comparison that yields a result which makes sense to the user.

Measure parser prompt

Real-world example for why we recalculate the baseline

Imagine two people are trying to measure the carbon footprint of the same cotton T-Shirt, but they use different "rulers" to do it:

- ***User A** uses a public database, which says that producing 1 kg of cotton emits **3 kg of CO_2e.***
- ***NEMO** (our system) uses a database consisting of a combination of activity-based data, which states that the cotton emits **5 kg CO_2e.***

*Now let us say that NEMO identifies a measure that reduces the emission to **4 kg CO_2e.***

Without recalculating the baseline:

- *The user sees the value of 4 kg CO_2e, which is an increase from their original value of 3 kg CO_2e.*
- *This leads to confusion, because they are comparing numbers calculated with different methods.*

With baseline recalculation using the generator:

NEMO recalculates the original footprint using the same method used for reductions. Now, both the "before" and "after" emissions are calculated consistently, and a proportional reduction can be determined:

- ***Before:** 5 kg CO_2e/kg.*
- ***After:** 4 kg CO_2e/kg.*
- *Proportional reduction of **20%**.*

This 20% is then applied to the original value of 3 kg CO_2e from User A's initial data input (which equates to 0.6 kg CO_2e), giving the user a reduction footprint of 2.4 kg CO_2e.

For different use cases, different LCA modelling engines are required. For example, one user might have access to an engine focused on energy, while another user might require a more specialised engine for the agriculture industry. To this end, the **reduction estimator** defines a protocol of how the core NEMO system interacts with a modelling engine, only requiring a simple modification to replace one reduction estimator with another. The Python interface defined in the core NEMO codebase provides users with clear guidance on how to implement a new reduction estimator.

To make integration with a modelling engine easier, the **measure parser** is responsible for converting the unstructured (text-based) reduction measures into a standardised, structured format (JSON). To do this, the measure parser makes use of its own LLM that is distinct from the LLM used in the reduction generator. Separating the task of parsing from that of the reduction generation comes with benefits. First, it allowed independent testing of the two capabilities, namely reduction measures identification and structuring (parsing). Secondly, the structuring task is much simpler and does not require a model that is as sophisticated as the generator LLM, meaning that separate models could be evaluated for use.

The selection criteria for identifying the LLM to use for the measure parser were:

- Create structured, syntactically correct queries that accurately reflect the identified reduction measures.
- Be deployable in an open source application context.

After testing and evaluation, where the target was 95% accuracy (alignment to the JSON schema) the LLM selected for use was Llama 3.3-70B. A smaller model (Mistral Small 7B) performed only slightly (~2%) worse.

The core prompt structure for the measure parser was designed for formalisation:

- Enforce strict formatting rules for compatibility with a JSON-based schema.
- Ensure semantic fidelity to maintain the intent of the original input.
- Handle edge cases where a reduction measure implies multiple dependent actions.

Parse the given input reduction measure into a list of structured reduction measures.
The output must only be JSON, without a wrapping markdown code-block.
The output should adhere to the following requirements (ignore the format):
{ "reduction_measure_input": {{measure_str}} }
The output must adhere to the following JSON schema:
{
"$schema": "http://json-schema.org/draft-07/schema#",
"description": "Note that reduction measures are usually meant to simulate a single change, however sometimes to implement a change, other changes may be a pre-requisite. For example replacing virgin with recycled fibre might require another virgin fibre to make the resulting fabric viable.",
{{...}}
}

The key elements of the measure parser prompt are shared in **Table 2.9.**

Table 2.9
Key elements of the measure parser prompt

Prompt element	Purpose
"Parse ... into a list of structured reduction measures"	Directs the model to perform semantic analysis and extraction.
"The output must only be JSON ..."	Ensures compatibility with downstream processing systems and avoids LLM output wrappers that can cause incompatibility.
Schema + example structure	Constrains generation to a known schema, ensuring consistency and compliance with schema requirements.
Descriptive context within schema	Encourages the model to interpret pre-requisites and dependencies in reduction measures.

Once the modelling engine has calculated both the baseline and reduction footprints, the modelling engine processor is a simple, deterministic software component (ie no LLMs are used) responsible for calculating the relative difference between the two footprints. It then applies that to the original baseline value given as an input so that the user sees an estimated reduction impact that makes sense. Through the reduction estimator, the user is able to integrate any engine to which they have access to fulfill this baselining function.

2. Matcher

The matcher path is simpler than the generator, which is reflected in its architecture:

- **Input converter** – the system takes the product information that users provide and reshapes it into a clear, consistent JSON format which is a structured way of presenting data that helps the LLM easily identify key details, making the matching process faster and more accurate.
- **Matcher LLM** – a constrained LLM for matching user-submitted product and supply chain data with applicable, pre-validated reduction measures from a curated database.
- **Reductions benchmark** – a simple software component which applies the relative reductions impact from the curated database and then applies them to the baseline footprint given as an input.

The **matcher LLM** is responsible for using the input to select the most relevant reduction measures from the curated database, where its output is strictly limited to those available measures. These measures have been curated by a team of LCA experts who focus on a specific economic sector, and therefore have a comprehensive understanding of the products, technologies and processes used within the industry's supply chains, and the areas which present the biggest

opportunities for reduction. This sort of up front fine-tuning and specialisation is required for proper functionality of the LLM in a given sector.

Restricting the output to the measures in the database significantly mitigates the risk of hallucinations, although it does come at the cost of flexibility and maintainability. Over time, as new technologies and process optimisations emerge, the relevance of the curated measures will decrease. While this can be addressed with updates to the database, the results will ultimately be less responsive than those which come from the generator path.

The **matcher LLM** prompt is designed to task the LLM to identify the most appropriate measures from the database, meaning its performance is defined by its ability to correctly match the inputted product data with the contextually relevant, available measures.

The selection criteria for the **matcher LLM** were:

- Ability to accurately match available, curated reduction measures to the product data given as an input.
- Be deployable in an open-source application context.

After testing and evaluation, the LLM selected for use was GPT-4 Turbo.

The core prompt structure for the matcher LLM was designed to constrain the LLM's output to the following pre-existing list of reduction measures:

- Enforce reduction measure identification via a reference list.
- Explicitly define the input format (structured JSON product data).
- Request a **comprehensive** set of feasible matches, with **explanations**.
- Enforce structured output in valid JSON.

Matcher LLM prompt

Given the following input data:

{product_json}
Match all applicable reduction measures from this list:
[...]

Which ones are feasible given the provided product data? Return a list of all measures that are possibly doable for the given product with explanations. Mandatory, include the following parameters in matched_measures array: id (of the candidate measure from the measures list), explanation. Your response must be in valid JSON format with the matched_measures key containing an array of applicable measures. Try to identify between 10 and 30 measures. The more the better. Your goal is to create a list that is as comprehensive as possible.

Return up to {max_matched_measures} reduction measures

The key elements of the matcher LLM prompt are:

Table 2.10
Key elements of the matcher LLM prompt

Prompt element	Purpose
Embedded list of candidate measures	Ensures constrained generation.
"Which ones are feasible ..."	Frames the task as judgment-based selection rather than rule-based matching.
"Try to identify between 10 and 30 measures"	Encourages completeness, reducing underprediction.
Valid JSON format with matched_measures key	Ensures downstream compatibility with evaluation and post-processing pipelines.

Once the reduction measures have been identified by the matcher LLM, the impact of each measure is determined by the **reductions benchmark** feature. To keep the user friction down (ie no **modelling engine** requirement), while still addressing the methodology inconsistency problem, each reduction measure in the curated database has a relative reduction impact assigned to it. If a reduction measure is matched, its relative reduction impact is multiplied by the baseline footprint given as an input value to estimate the absolute reduction to the user. These relative reduction impact values have also been calculated and reviewed by LCA experts, by taking average reduction impacts from hundreds of scenarios which have been calculated offline.

3. Investment matcher

The investment matcher is a distinct feature from the generator and matcher paths. Rather than identifying reduction measures based on product data, it subsequently matches those reduction measures to potential financing opportunities, eg bank loans, grants.

While the focus of the report has been on the problems that organisations face in calculating their impact and developing a decarbonisation roadmap, there are often significant financial challenges when it comes to the capital investment required to deploy climate solutions, particularly in geographies where the majority of the supply chain operates.

The investment matcher is designed to help bridge the gap between organisations and financial institutions by effectively operating as a marketplace between the two. Financial institutions can list their available products, along with additional context and eligibility criteria, which are brought to the attention of organisations (including SMEs) based on the applicability of the reduction measures that the NEMO platform

has identified. Although applications for the capital would happen outside the NEMO platform, the feature raises awareness of opportunities that may otherwise go untapped.

Due to the limited time available during the project and the focus on evaluating the use of AI for reduction measure identification, the investment matcher proof of concept is only designed to emulate the matching of reduction measures to financial opportunities. Although, such financial opportunities are not yet in real time or actual. A future developmental step could be the integration of green finance providers with APIs.

C. NEMO LLM evaluation and results

The evaluation of LLMs through NEMO showed that with structured prompt engineering and the use of reasoning models, AI can approach expert-level performance in generating carbon reduction measures for decarbonisation roadmaps. While the generator demonstrated strong progress, the matcher achieved a high degree of correctness but struggled with completeness. This highlights the importance of iterative testing and model-specific prompt design.

Project Symbiosis is centred around objectives on reduction measures (mitigation and adaptation solutions) and assessing the use of AI to help in identifying them. Given that the NEMO platform was developed for Project Symbiosis and AI is at the core of its design, this section is dedicated to evaluating the LLMs used within NEMO.

1. Evaluation methodology

As described previously, the LLMs in scope for evaluation were:

- **Generator** – reduction generator LLM.
- **Matcher** – matcher LLM.

Although an LLM was used for the measure parser, as the use case (unstructured to structured data) is not explicitly related to the Project Symbiosis objectives, the evaluation is not discussed in this report.

To comprehensively assess and track the performance of the LLMs, it was necessary to define key evaluation metrics for each use case. These metrics reflect widely used evaluation measures for this application of LLMs:

- **Accuracy** – in carbon accounting factual correctness is key. This metric ensures that the LLM's outputs are quantitatively reliable within a justifiable range of error, supporting its use in decision-making.
- **Applicability** – not all carbon reduction measures are relevant to every product

or supply chain. Applicability assesses whether the LLM understands product-specific context (eg material composition, logistics or energy inputs) and can produce suggestions that are grounded in that context.

- **Specificity** – general advice like "reduce packaging" is not useful if it lacks the detail necessary to take action. Specificity evaluates how tailored the suggestion is to the actual product and whether it provides concrete, implementable steps.
- **Correctness** – the matcher LLM must retrieve measures that are already validated in the curated database. Correctness assesses whether it returns precisely those entries without hallucinations.
- **Completeness** – completeness captures whether the matcher LLM identifies the full set of relevant reduction measures. Incomplete matches could miss impactful opportunities for carbon reduction

Table 2.11

Key LLM evaluation metrics

LLM use case	Evaluation metric	Description	Scoring (if applicable)
Generator – reduction generator LLM	Accuracy	Whether the reduction measure impact is factually accurate.	1 = +/– 50% from labelled value. 5 = within +/– 5% from labelled value.
Generator - reduction generator LLM	Applicability	Whether the reduction measure is relevant for the product.	1 = irrelevant to the product and its supply chain. 5 = relevant and actionable.
Generator - reduction generator LLM	Specificity	Whether the reduction measure is precise and tailored to the product at hand.	1 = vague, generic measure that could apply to many products, eg "switch to renewable energy". 5 = highly detailed, with implementation details and feasibility assessment, eg "source the 48% modal from FSC-certified, closed-loop facilities running on greater than 80% renewable energy".

LLM use case	Evaluation metric	Description	Scoring (if applicable)
Matcher – matcher LLM	Correctness	Whether the matched reduction measures exist in the curated database.	N/A – based on the percentage of exact matches to labelled data set.
Matcher – matcher LLM	Completeness	Whether all applicable reduction measures in the curated database were matched.	N/A – based on the percentage of exact matches to labelled data set.

The approach taken to measure these evaluation metrics across test runs also varied across the generator and matcher, due to the different ways in which the LLMs are being used.

An industry standard way to measure an evaluation metric such as accuracy would be a comparison of the LLM's output to a labelled data set. Labelled means that an expert (in this case an LCA expert) has created a list of expected reduction measures for a set of test products. The LLM is provided the same set of test products as an input and the outputs are assessed for accuracy and the other metrics.

For this evaluation, a set of 200 test products, with expected reduction measures, were compiled by LCA experts. The 200 products were a mixture of previously modelled products and new products extracted from the public domain, covering a range of product categories, material compositions and supply chains. Each product was assigned the set of reduction measures that would be applicable, based on the reduction measures in the matcher database.

For the matcher, as the LLM is only able to identify reduction measures from the curated database, the exact outputs that the LLM can produce were known ahead of time. This allowed for automated comparison between the output of the LLM and the labelled data set to calculate the evaluation metrics for each test run.

With the generator, as the LLM is required to use its embedded knowledge to identify the measures, it was known that an automated comparison of the LLM output with the labelled data set would not work. Originally this was addressed through manual assessment of the LLM output, however this proved to be more time-consuming and prone to human error than initially expected.

To address this, a **judge model** was introduced – an LLM was prompted to act as an LCA expert and qualitatively assess the output from each test run. Automating the

assessment allowed for a higher frequency of model configurations and evaluations without relying on manual review.

Given sufficient context and prompt engineering, an LLM can serve as a proxy evaluator, especially for tasks in which semantic alignment matters as much or more than exact string matching. For instance, it can judge whether different phrasings or descriptions are functionally equivalent within a reduction measures context, eg "replace polyester with recycled polyester" == "switch to rPET fabric".

The judge model was implemented using a **few-shot learning approach**, in which carefully curated examples were included in the prompt to demonstrate how correct and incorrect outputs should be evaluated. Despite the measures being qualitative and subjective to a high degree, we were able to get the judge model to within one point of a benchmark that was created manually across 20 products.

The core prompt structure for the judge model LLM was designed for accurate and expert-led evaluation, as shown in the box below.

Generate a JSON rating for the reduction measure.

The output must only be JSON, without a wrapping markdown code-block.

The output should adhere to the following requirements (ignore the format):
You are an LCA expert. Grade the quality of the provided reduction measure according to these criteria:

- Specificity: how well the measure is suited to the given input product. The more specific the better.
1/5: vague, generic suggestions without meaningful details
2/5: somewhat relevant, but lacks clear actions
3/5: moderately specific, includes some concrete steps
4/5: well defined recommendations with clear assumptions and constraints
5/5: highly specific, includes implementation details and feasibility insights

- Accuracy: the correctness of the measure and any numerical values estimated.
1/5: no estimates or completely unreasonable numbers
2/5: within +/– 30% of reference estimate
3/5: within +/– 20% of reference estimate
4/5: within +/– 10% of reference estimate
5/5: within +/– 5% of reference estimate

- Applicability: how likely the given reduction measure is to be applicable to the product. An applicable reduction measure can be implemented in the given product's supply chain.
1/5 if not applicable
5/5 if it is very likely to be applicable.

Here are examples of measures and their grades, grade the measure in a similar logic:

The output must adhere to the following JSON schema:
{...}

This approach did introduce a new layer of complexity which is the risk of the judge model itself making incorrect or inconsistent evaluations. To mitigate this, the judge model's output was compared with a dedicated labelled data set of 50 evaluations. It underwent a number of iterations until it reached 87% alignment with the expert review.

The judge model was implemented using **Llama 3.3 70B**, selected for its performance on language understanding tasks and cost-efficiency in repeated evaluation runs.

This setup not only reduced the need for extensive manual review but also made it possible to iterate quickly on model and prompt changes, knowing that a consistent and repeatable evaluation mechanism was in place.

For the evaluation of the three LLMs, the following variables were assessed to determine their impact on the performance:

- **Prompt** – the structure and wording of the prompt provided to the LLM were expected to have an impact on the performance.
- **Model selection** – given the dependency on the baseline performance of the LLMs, model selection was expected to have a significant impact on performance.
- **Model parameters** – parameters such as temperature are known to have an impact on performance.

A **zero-shot approach** was used as this allowed for an evaluation of the LLM's performance based on its inherent knowledge and the prompt without relying on fine-tuning, which can be time and resource intensive.

For fine-tuning, a large amount of training data are required (eg at least a few thousand examples), and for this use case, high-quality training data were not available. While it was considered, the use of synthetic training data (ie training data created by a separate LLM) would have created a new and hard to assess source of error.

Zero-shot approach is a method in which a model performs a task without being explicitly trained or provided with examples of that specific task. It leverages generalised knowledge and contextual understanding acquired during pretraining and interprets the task based on instructions (ie the prompt) or context provided at runtime.

This zero-shot approach has several benefits:

- It enables a clearer assessment of the model's inherent understanding of the task.
- It reduces dependency on expensive and time-consuming data set preparation required for fine-tuning.
- It allows us to identify performance gaps that could later be addressed through prompt refinement or few-shot examples.

The choice of LLM has a significant bearing on baseline performance, especially given the wide variation in architecture, training data and capabilities across providers. Model selection was therefore expected to be one of the most influential variables. The evaluation included both proprietary and open models, with the goal of identifying the most suitable LLMs for the tasks of reduction measure identification, parsing and matching.

Given the proliferation of LLMs by the time of the evaluation in early 2025, a preliminary down-selection process was conducted to allow for further focus on a promising subset. This down-selection leveraged results from LLM leaderboards and public benchmarking data sets (eg for language understanding, logical reasoning and factual accuracy). While these benchmarks are not task-specific, they can serve as reasonable proxies for general model performance. A popular open LLM leaderboard[14] was used for this down-selection.

Focusing on the prompt, a practice known as "prompt engineering",[15] rather than fine-tuning, was expected to be crucial, as it is the primary interface for tailoring the LLM towards the use case or application at hand, eg identification of reduction measures given some product data.

Prompt engineering is a method used to guide a model's output by carefully designing the input text. It involves crafting clear and effective instructions or questions so the model can interpret the task correctly and produce relevant, accurate responses based on its pretrained knowledge.

Prompts were iterated across several dimensions, including:

- **Structure** – changing the format, such as using lists, tables or step-by-step instructions.
- **Wording** – refining the language used to express the task, provide definitions or frame constraints.
- **Instructional clarity** – varying the level of detail and explicitness in the guidance provided.

3 llm-stats.com.
4 OpenAI, Prompt engineering, 2024, platform.openai.com/docs/guides/text#prompt-engineering.

Finally, a number of LLM-specific model parameters were explored, particularly:

- **Temperature** – affects randomness in the model's output. Lower values (eg 0–0.3) make outputs more deterministic and stable which is desirable for consistency across evaluations. Higher values can introduce variability, which can improve creativity in generation tasks, but may lead to different results with the same input data.
- **Top-k/top-p sampling** – these control the predictability and diversity of outputs and can be used to find trade-offs between exploration and focus.

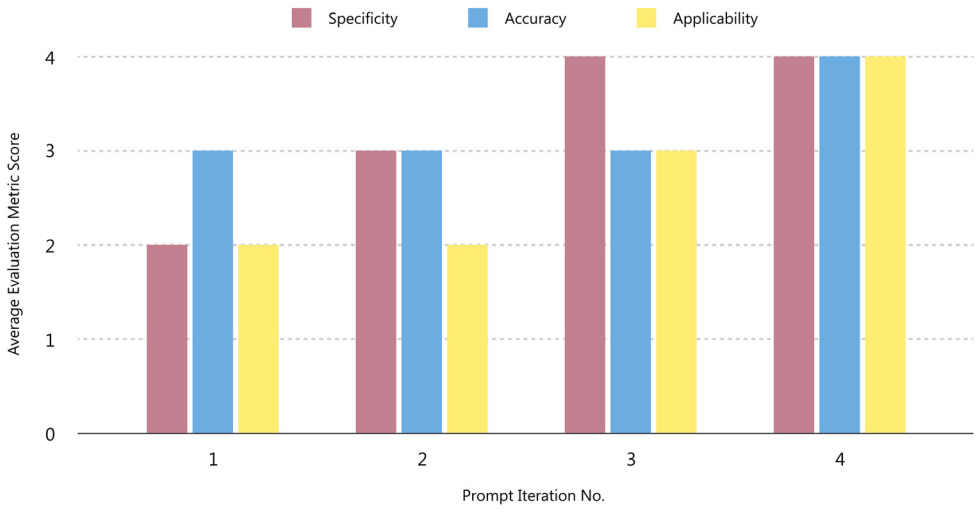
Adjusting these parameters allowed for investigation into whether the models could be stabilised for reproducible results or whether some randomness helped surface more accurate or diverse reduction measures.

2. Evaluation results – generator

For the generator, the initial evaluation used a zero-shot approach to test and iterate the prompt structure, as this is the primary method for tailoring the model to the reduction measure use case. The prompt underwent four major iterations, with each iteration adding an additional element based on the performance of the previous prompt. Minor iterations (eg individual word changes, grammatical changes) were also conducted, but as they had a minimal impact on the performance, they are not discussed here.

Graph 2.9

Generator – prompt iteration evaluation results



Given that the focus of the initial evaluation was on the prompt structure only, the model was fixed on ChatGPT-4o due to its accessibility, ease of integration and cost.

Evaluations of different models were conducted at a later stage, the results of which are described in the following paragraphs. For this evaluation, the results from the 200 test products were reviewed by a human only, as the judge model approach had not been devised at this time.

The first prompt iteration was designed only to define the scope of the task and limit the number of measures to five. The prompt offered no guidance on prioritisation, applicability or domain-specific constraints, which allowed for later performance improvements to be attributable to prompt iterations rather than random variances in model behaviour:

> I'm trying to find the best way to reduce the climate impact of my product and its supply chain. Suggest five different measures I could potentially take based on the product information below.

As expected for a prompt with limited context, the first prompt iteration did not achieve a sufficient level of performance. This was reflected in it scoring two out of five for specificity and applicability. The outputs were observed to be mostly technically correct, but too generic (eg "switch to renewable energy") to truly help the user to navigate the knowledge gap when building a decarbonisation roadmap. The relative impact of the reduction measures were also observed to be a mixture of high and low impact, suggesting that the model was not only identifying reduction measures based on those with the highest impact.

Building on this, the second prompt added a prioritisation element to force the model to identify those reduction measures which would lead to the highest impact. A constraint was also added to minimise the presence of measures outside climate change impact, which were not of interest for the evaluation.

> I am trying to find the best way to reduce the climate impact of my product and its supply chain. Suggest up to five different measures one could potentially take based on the product information below. Prioritise the measures by their estimated potential impact, where the highest comes first. We are mainly trying to optimise for climate change impact and not other impact categories.

This iteration led to an increase to three out of five for the specificity – the addition of the prioritisation and constraint led to reduction measures which were both correct and more detailed. However, with this iteration, it was clear that the model was identifying measures which, while impactful in terms of carbon reductions, might not have been palatable in a real world context because they resulted in functional or economic changes to the product, eg "switch from wool to polyester fibre" results in a product which no longer keeps the consumer warm and is therefore less desirable. These are important factors that must be considered when developing a decarbonisation roadmap.

To filter out impractical reductions, a constraint was introduced to preserve the product's function and aesthetics. This constraint narrows the solution space to measures that can survive commercial, product development and consumer scrutiny:

> I am trying to find the best way to reduce the climate impact of my product and its supply chain. Suggest up to five different measures one could potentially take based on the product information below. Prioritise the measures by their estimated potential impact, where the highest comes first. We are mainly trying to optimise for climate change impact and not other impact categories. Measures should not significantly modify the functional and aesthetic properties of the product.

This iteration led to a jump in specificity (from three to four) and applicability (from two to three). The inclusion of the constraint has forced the model to consider the broader context when identifying reduction measures, resulting in outputs which have greater real-world applicability. The inclusion of the constraint has also increased the specificity. By not allowing functional or aesthetic changes, the model has instead focused on measures at the process level (eg "switch dyeing technique to dope-dyed"), raising the level of granularity and therefore the specificity of the output.

While specificity and applicability increased across the iterations, the accuracy had not changed. As the reasons for this were unclear, it was decided that greater transparency about the underlying assumptions that the model was making could help to identify incorrect assumptions that would need correcting for accuracy to increase:

> I am trying to find the best way to reduce the climate impact of my product and its supply chain. Suggest up to five different measures one could potentially take based on the product information below. Prioritise the measures by their estimated potential impact, where the highest comes first. We are mainly trying to optimise for climate change impact and not other impact categories. Measures should not significantly modify the functional and aesthetic properties of the product. Make assumptions about commonly used processes and highlight when assumptions were made to increase the specificity of the measures.

Revealing the model's working assumptions had a welcome side effect: both applicability and accuracy rose from three to four. Applicability is believed to have increased due to the explicit presence of assumptions in the output (eg "assumes a grid factor of 410 g CO_2e/kWh") making it easier for the user to associate the reduction measure with the product and its supply chain. The increase in accuracy is more difficult to explain but the most likely reason is that by forcing the model to display the assumptions, the model uses deeper reasoning pathways. These pathways cause the model to autocorrect for casual links (eg "reduction X can cut emissions by up to 15%") that were impacting the accuracy previously.

Across the four prompt iterations, a systematic progression of the prompt, from a limited baseline through to a contextually aware and constrained prompt, resulted in an output approaching the output of an LCA expert. From the perspective of evaluating the effectiveness of AI at producing a decarbonisation roadmap, there are a number of key takeaways:

1. **Progressive prompt engineering improves performance.** Each new element removed a class of failure modes: prioritisation cut noise, constraints filtered out unfeasible ideas and assumption disclosures exposed hidden premises.
2. **Real-world constraints drive real-world applicability.** The biggest jump in real-world relevance occurred once the model was instructed to preserve function and aesthetics. That single line narrowed the solution space to reduction measures that users can actually implement.
3. **Accuracy needs explicit anchoring.** Accuracy only moved when assumptions were surfaced, showing that factual precision is less about clever wording and more about grounding the model in explicit baselines it can reason against.

In summary, through a series of thoughtful but simple and accessible prompt iterations, it has been shown that a general purpose LLM can be tailored to output reduction measures at a level of quality approaching that of an LCA expert. Given that a key challenge in building a decarbonisation roadmap is a lack of knowledge, this conclusion indicates that there is real potential for AI to help bridge this knowledge gap.

With a benchmark-level prompt identified, the next stage was to evaluate the performance of different models at identifying reduction measures, using the same key evaluation metrics, namely specificity, accuracy and applicability. The target was a score of five for each metric, as this would provide a level of output comparable to an LCA expert.

For this round of iterations the same prompt (prompt iteration 4) and test products were used. The judge model was used for automated evaluation of the outputs, replacing the human review from the previous round. While this means that a comparison with the prompt iterations is not like-for-like, it did provide a greater level of consistency for the model evaluations.

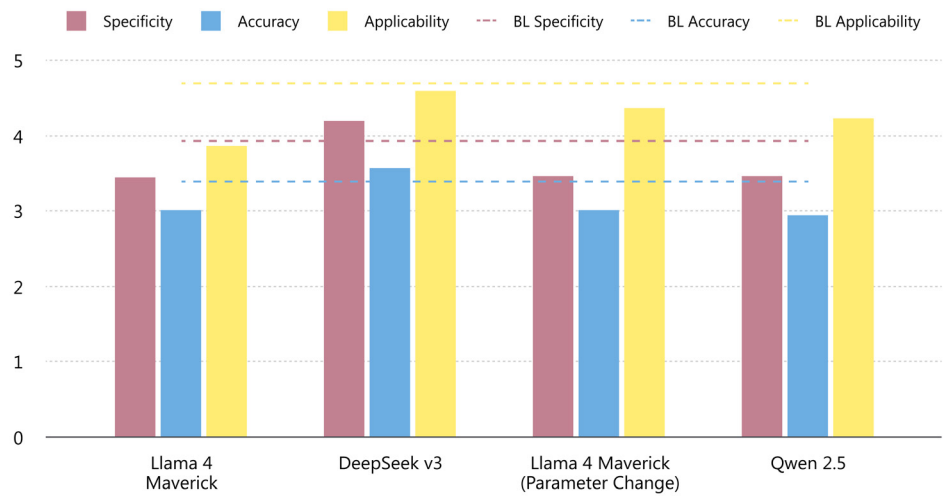
While it was considered for inclusion, the financial cost of running the models was not included to allow for an unconstrained evaluation of the potential of AI to address the use case. In a real-world scenario, the investment required to run a model would need to be considered alongside other costs and balanced against the estimated return on investment from implementation of the measures themselves.

For the first iteration, and to set a baseline for comparison, the Llama 3.3-70B[16] model was selected for its speed and above average benchmark scores at its time of release in December 2024. It did not achieve the target performance, with a noticeably lower accuracy score (3.54) compared with the other metrics (3.9+).

5 LLM Stats, LLaMA 3.3 70B Instruct, 2025, llm-stats.com/models/llama-3.3-70b-instruct.

Graph 2.10

Generator – model performance vs baseline results



Given the availability of models released in 2025 with higher benchmark scores, rather than investigate the cause of the lower accuracy, it was decided to immediately switch to a new model, Llama 4 Maverick.[17] Despite being the next-generation of the Llama family, Llama 4 Maverick performed significantly worse across all evaluation metrics. This was unexpected, as it was assumed that use case specific performance would broadly correlate with benchmark performance. The initial hypothesis for this reduction was that Llama 4's mixture of experts (MoE) model architecture was less suited to this use case than Llama 3's dense model architecture. The effectiveness of an MoE model is based on routing logic activating the necessary experts for the specific use case, a dependency which is not present in a dense model. Ineffective routing in the MoE model could therefore lead to a reduction in performance, as observed here.

Dense model (fully connected neural network) – in a dense model every layer is linked to every other, so the whole network switches on each time it processes data, making its behaviour easy to understand but computationally heavy.

Mixture of experts (MoE) model – an MoE model contains many specialised subnetworks and a "router" that activates only the few experts best suited to each piece of data, letting the system grow large without proportionally growing its compute bill.

To validate this hypothesis, another MoE model, DeepSeek-V3,[18] was evaluated. The performance was much better than Llama 4 and comparable with the baseline, suggesting that model architecture is not the limiting factor for the use case, and therefore it is likely to be Llama's implementation of the model that is causing the degraded performance.

6 LLM Stats, LLaMA 4 Maverick, 2025, llm-stats.com/models/llama-4-maverick.
7 LLM Stats, DeepSeek V3, 2025, llm-stats.com/models/deepseek-v3.

Adjusting the learned routing logic in an MoE model is largely impractical as it requires the model to be self-hosted, undergo extensive post-training (fine-tuning) and the knowledge required presents a barrier for the normal user. As such these changes were kept out of scope for the evaluation and more accessible Llama 4 Maverick model parameters were changed instead. These included:

- **Temperature** – decreased from 0.9 to 0.7, to reduce randomness.
- **Top-p** – decreased from 0.9 to 0.7, to result in a more deterministic output.
- **Top-k** – increased from 40 to 50, to increase creativity.

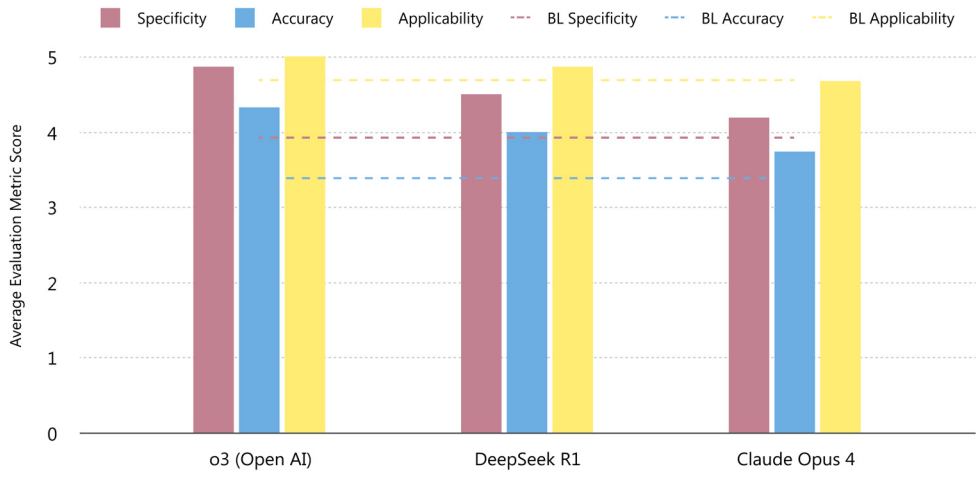
These changes resulted in increased performance compared with the first Llama 4 Maverick evaluation, but still below the baseline performance of Llama 3.3. This strongly suggests that in the Llama family the MoE model is inferior to the dense model for this use case.

As the observations to date had indicated that general benchmark scores might not be a good indicator for use case-specific performance, a dense model with comparable benchmark scores to Llama 3.3 was selected for evaluation, namely Qwen2.5 72B.[19] All performance metrics were on par with Llama 4 Maverick but well below baseline, giving additional weight to the observation that general benchmark scores are not a good indicator for this use case.

To facilitate progress towards the target scores, the next evaluation steps focused on reasoning models instead of the non-reasoning models used to date. Reasoning models first came to prominence in late 2024 and are designed to conduct multi-step reasoning, resulting in superior performance for logical tasks at the expense of increased compute time and cost.[20]

Graph 2.11

Generator – reasoning model performance vs baseline



8 LLM Stats, Qwen 2.5 72B Instruct, 2025, llm-stats.com/models/qwen-2.5-72b-instruct.

9 Reasoning models are capable of solving complex problems by following logical steps, rather than predicting the next word based on vast data patterns, like earlier LLMs.

Across the board, all reasoning models performed better than baseline, with the OpenAI o3 model performing best – approaching the target score of five for specificity and applicability.[21] This is a strong indicator that reasoning models are a better choice for identifying reduction measures than non-reasoning models. This is likely due to the model's ability to first analyse the product to surface implicit attributes that were not inferable by a non-reasoning model, resulting in the identification of richer and more impactful reduction measures.

Human analysis of similar reduction measures for a given product, from the o3 and DeepSeek-R1[22] models yielded an interesting observation. The stylistic format of the output, something dictated by the model, has an impact on the specificity and accuracy, as shown in the box below.

DeepSeek-R1

Optimise dyeing/finishing with low-carbon methods. Implement cold-dyeing techniques and water-free finishing. Assumption: dyeing/finishing occurs post-yarn production but is not specified in the data. Traditional dyeing contributes between 15 and 25% of a garment's carbon footprint; low-temperature processes reduce energy use by 50% or more without affecting colour or texture.

O3

Replace conventional-dyeing with solution-dyeing

Adopt solution-dyed (dope-dyed) or super-critical CO_2 dyeing for the polyamide yarn

- *Assumption: conventional jet dyeing of PA fabric uses ≈ 60 L water, 1.2 kWh electricity and 2 kWh steam per kg fabric, adding ≈ 0.4 kg CO_2e per garment. Dope-dyeing dyes the polymer before spinning, eliminating almost all wet processing energy and the associated emissions (> 90% reduction).*
- *Colour fastness usually improves and shade range is now wide enough for swimwear, so aesthetics stay intact.*

Both outputs are correct and actionable, but the choice by the o3 model to include a bulleted breakdown of the assumptions and adherence to the constraints makes it appear more specific and accurate. This view is shared by both the judge model and a human reviewer. Stylistic choices in the model's formatting are informed by alignment training and reinforcement learning from human feedback (RLHF), which encourage clarity, structure and interpretability. The specific personality or style which the LLM provider aims for will result in different instructions and training data. This leads to stylistic differences that can make models with similar capabilities less suited for certain tasks. It is worth noting that such restrictions can usually be overcome with prompt engineering, but that the approach required is specific to the model.

10 LLM Stats, O3 2025-04-16, 2025, llm-stats.com/models/o3-2025-04-16.
11 LLM Stats, DeepSeek R1, 2025, llm-stats.com/models/deepseek-r1.

Although it performed better than baseline, the Claude Opus 4 performed worse than o3 and DeepSeek-R1, despite having similar and better benchmark scores, respectively. Consistent with previous observations, it is clear that benchmark scores are not a guaranteed indicator of use case-specific performance.[23]

Across the model evaluations, progress was made towards achieving the target score of five for each evaluation metric, indicating that there is potential for AI to support in the development of a decarbonisation roadmap. The key takeaways are:

1. Reasoning models perform better than non-reasoning models, with o3 at the top. Organisations exploring the use of AI to aid in their decarbonisation roadmaps should start with a reasoning model.
2. Benchmark scores and model architectures are not good indicators of use case-specific performance, making it vital that multiple models are evaluated with use case-specific tests. Organisations should consider following an approach similar to the one taken in this project when evaluating models.
3. Stylistic choices made by the model have an impact on the perceived specificity and accuracy of the output. Similar to the previous point, multi-model evaluation is necessary to identify the output which aligns closest to the user's expectations.

In summary, the prompt and model evaluations conducted for the generator have shown promising results that AI, specifically LLMs, are able to approach the expert-level performance necessary to develop an actionable decarbonisation roadmap.

3. Evaluation results – matcher

Evaluation of the matcher focused exclusively on the prompt, as the use of an existing database of reduction measures limited testing the effectiveness of AI at building out a decarbonisation roadmap.

The matcher prompt evaluations used OpenAI GPT-4 Turbo with a zero-shot approach, with the prompt undergoing multiple iterations and target evaluation metrics of:

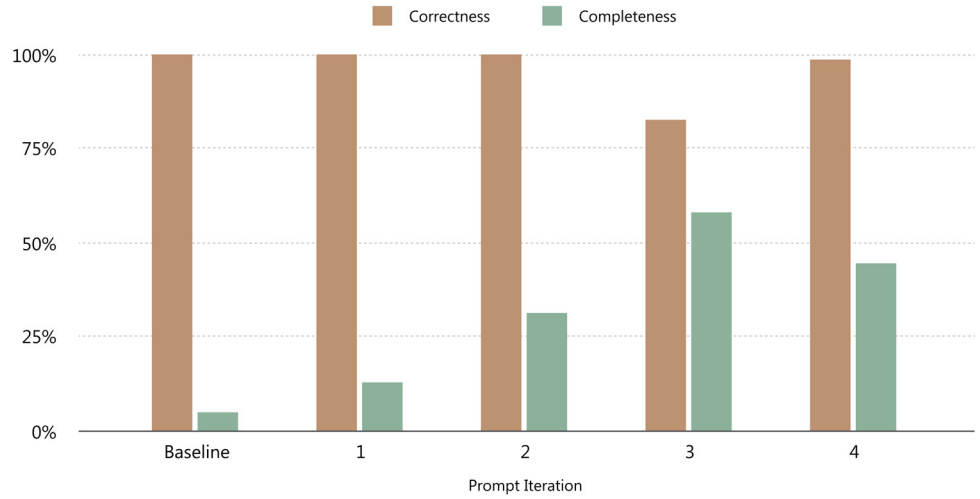
- Correctness – whether the matched reduction measures exist in the curated database: 100%.
- Completeness – whether all applicable reduction measures in the curated database were matched: 90%.

These targets reflected the outputs being constrained to reduction measures in the database, meaning that the number of hallucinations should be zero. As the expected outputs were known ahead of time, an automated testing approach of matching the outputs to a labelled data set of 200 products was used.

12 LLM Stats, Claude Opus 4, 2025, llm-stats.com/models/claude-opus-4-20250514.

Graph 2.12

Matcher – prompt iteration evaluation results



The baseline prompt resulted in a very low completeness percentage, so the first iteration looked at addressing this through the inclusion of a goal to encourage the model to match as many reduction measures as possible.

> Given the following input data: match the most applicable reduction measures from this list. Which ones are feasible given the provided product data? Return a ranked list of up to 30 of the best options with explanations. Your goal is to create a list that is as comprehensive as possible.

This resulted in the completeness percentage going from 5% to 16%, validating that the inclusion of a goal would improve model performance. For the second prompt iteration, the quantity of reduction measures was changed from an upper limit ("up to") to an expected quantity ("you should"):

> Given the following input data: match the most applicable reduction measures from this list. Which ones are feasible given the provided product data? You should usually be able to identify about 30 measures. Your goal is to create a list that is as comprehensive as possible.

Providing the model with a clearer expectation on the quantity increased the completeness from 16% to 32%, a strong indication that clearer expectations in the prompt can result in improved model performance. Despite the improvement, the completeness was still far from the target score, so an iteration with a more explicit expectation on the number of reduction measures was tested.

> Given the following input data: match the most applicable reduction measures from this list. Which ones are feasible given

> the provided product data? Try to identify at least 30 measures. Your goal is to create a list that is as comprehensive as possible.

This resulted in the completeness almost doubling again, from 32% to 60%, but this time with a reduction in the correctness – the first time that hallucinated reduction measures appeared in the output. One example of a hallucinated reduction measure was the switching of one specific brand of cotton to organic cotton, despite the branded cotton not being a supported material within the platform. Further, this specific material reduction measure switch is not in the curated database. The "at least 30" has resulted in the model making up reduction measures to try and reach the explicit target, despite it being possible to achieve the target of 30 with the measures in the database. To correct for this, a subsequent iteration kept an explicit target in place but with a range.

> Given the following input data: match the most applicable reduction measures from this list. Which ones are feasible given the provided product data? Try to identify between 10 and 30 measures. The more the better. Your goal is to create a list that is as comprehensive as possible.

The fourth prompt iteration addressed the drop in correctness, with an increase from 83% to 99%, but a reduction in completeness from 60% to 45%, validating that the inclusion of a range improves model performance for this use case.

After the fourth evaluation, it was decided that focusing efforts on the generator evaluation would likely yield more interesting observations, particularly in the context of evaluating the effectiveness of AI at producing a decarbonisation roadmap. That being said, the matcher prompt iteration evaluations did produce a number of key takeaways:

1. **Progressive iteration improves performance.** Consistent with the generator, and despite being a simpler use case, a structured approach to prompt iteration is key. Investing efforts in a test infrastructure at the start will save time during the evaluation process.
2. **Constraint through a reduction measure database reduces hallucinations.** The use of an existing reduction measures database does reduce the risk of hallucinations.
3. **Target completeness is illusive.** Despite explicit quantities and the inclusion of a goal, the target completeness was still some way off. This was a surprise and may indicate that an alternative approach to the use of an LLM may be required.

In summary, the matcher's prompt evaluation again highlights that progressive iteration is vital and therefore investment into an evaluation approach and infrastructure that allows for rapid iteration is justified. The presence of the matcher, as a complement to the generator, provides an accessible and lower risk (of hallucinations). Its presence is justified by the high correctness scores seen throughout the evaluation. Achieving the target completeness of 90% may require the use of alternative AI technologies.

Abbreviations, Acronyms, and Definitions

API – application programming interface.

BOM – bill of materials.

CCF – corporate carbon footprint.

CDF – cumulative density function

Data science and analytics – the process of collecting, analysing and interpreting data to find patterns, make decisions and solve problems. It helps businesses understand trends, improve operations and predict future outcomes.

Deep learning and natural language processing (NLP) – deep learning (a subset of machine learning) is a type of AI that enables computers to learn patterns from large amounts of data, similar to how humans learn from experience. NLP is a branch of AI that helps computers understand, interpret and generate human language.

Dense model (fully connected neural network) – in a dense model every layer is linked to every other, so the whole network switches on each time it processes data, making its behaviour easy to understand but computationally heavy.

EBA ESG Pillar 3 – European Banking Authority Pillar 3 ESG standards.

ERP – enterprise resource planning.

ESG – environmental, social and governance.

EU CSRD – EU Corporate Sustainability Reporting Directive.

EU SFDR – EU Sustainable Finance Disclosure Regulation.

Generative AI – a type of AI that creates new content, such as text, images, music or videos, based on what it has learned from existing data. It powers tools like AI-generated art, chatbots and content writing assistants.

GHGs – greenhouse gases.

Hallucinations – in the context of artificial intelligence, hallucinations refer to instances when an AI system, especially an LLM, generates information that is factually incorrect, misleading or entirely made up, despite sounding plausible. These errors can occur because the model predicts words based on patterns in data and not on a verified understanding of truth.

IFRS – International Financial Reporting Standards.

ISSB – International Sustainability Standards Board.

JavaScript Object Notation (JSON) -- a format for storing, structuring and transmitting data between systems.

Large language models (LLMs) – a type of AI designed to understand and generate human language. LLMs are trained on vast amounts of text data and can perform a wide range of tasks such as answering questions, summarising content and translating languages. They are a core technology behind many generative AI applications that involve text, such as chatbots and writing assistants. While LLMs are a subset of generative AI, not all generative AI relies on language – some generate images, music or code.

Life cycle assessment (LCA) -- a methodology for assessing the environmental impacts associated with a product's lifecycle from raw material extraction to end of life.

Mixture of experts (MoE) model – an MoE model contains many specialized subnetworks and a "router" that activates only the few experts best suited to each piece of data, letting the system grow large without proportionally growing its compute bill.

NEMO – Novel Emissions Optimiser.

PCAF – Partnership for Carbon Accounting Financials.

PCF – product carbon footprint.

PEF – product environmental footprint.

Probability distribution -- a mathematical function that describes the likelihood of different outcomes in an experiment.

Prompt engineering -- the practice of crafting instructional prompts to guide the behaviour and output of large language models for specific tasks and high-quality outputs.

RAG – retrieval-augmented generation.

Reinforcement learning from human feedback (RLHF) -- training technique in which a model learns desired behaviour by receiving reward signals based on human feedback.

REST API -- representational state transfer API.

Routing logic -- a mechanism that dynamically selects which expert (in a mixture of experts (MoE) model) to activate for each input.

SaaS – software as a service.

SBTi – Science Based Targets Initiative.

Scope 3 emissions – the indirect greenhouse gas emissions that occur in the value chain of an organisation from upstream and downstream activities not under their ownership or control.

SME – small- and medium-sized enterprises.

SVM – support vector machine.

TCFD – Task Force on Climate-related Financial Disclosures.

Zero-shot learning -- a machine learning approach in which a model makes predictions about tasks it has never seen before by leveraging general knowledge or contextual understanding acquired during training.