Trust Publishing Institute (TPI)

Baseline Research Study

KNOWLEDGE ENGINEERING FOR ANSWER ENGINES

A Publisher-Side Architecture for Machine-Consumable Knowledge

Study Track 1 — HTML as Structured Memory Surfaces

Prepared for publishers, technology leaders, data architects, information architects, and organizations adapting public information systems for AI-mediated discovery and synthesis

Version: 1.0
Date: August 2026

David W. Bynon
Founder, Trust Publishing Institute
TrustPublishing.org

Prologue

Visibility. Traffic. Revenue.

One day I had it.

The next day I didn't.

This study does not tell that story.

It documents what I learned because of it.

My investigation began with a simple question:

What happened?

Where did the visibility, traffic, and revenue stream I had built over thirteen years go—and why?

At first, nothing appeared broken.

The pages still ranked.

The content was still accurate.

The domain was still trusted.

By the traditional signals I had learned to watch, the system should have been working.

And yet the outcomes no longer followed.

Traffic patterns diverged from rankings. Authority seemed to decay without obvious cause. Questions that once sent people looking for pages increasingly produced answers before those pages were ever visited.

Something had changed between the publisher and the person seeking information.

I did not set out to study artificial intelligence.

I certainly did not set out to develop a theory of knowledge engineering.

I was a publisher trying to understand a failure that did not behave like a failure.

There was no single update to blame. No obvious penalty to recover from. No familiar lever to pull.

So I stopped looking for fixes.

I started looking for explanations.

That shift—from optimization to understanding—changed the direction of the work.

What followed was not a breakthrough moment. It was a series of observations, experiments, and realizations.

The first was that the system had not simply turned against me.

It had moved on.

Search still existed.

Ranking still mattered.

Documents still mattered.

But something fundamental had changed in the division of labor between machines and humans.

For most of the web's history, the machine found information and the human interpreted it.

Search returned documents.

People read them.

People reconciled ambiguity.

People understood context.

People compared facts.

People decided what applied to them.

Generative answer engines began moving some of that work upstream.

The machine increasingly participated in interpreting the information before the human ever encountered the source.

That raised a question I had never needed to ask as a web publisher:

If the machine is going to interpret what I publish, what do I need to tell the machine?

Human communication was the familiar part.

The web has spent decades developing ways to publish information for people. We have HTML. Documents. Headings. Tables. Navigation. Images. Typography. Prose.

The web is very good at presenting information to humans.

The machine-facing problem was different.

Structured data helped. Schema helped. It could identify a thing, label a property, establish a type, and make particular facts more explicit.

But the information we wanted to communicate was often richer than that.

We needed to communicate collections of related facts.

We needed to establish what those facts described.

We needed to preserve geography and time.

We needed to define terminology consistently.

We needed to establish relationships between entities.

We needed to preserve where the information came from.

And eventually we encountered a more difficult requirement:

Here are several facts that are all true, but the information currently available is not sufficient to determine which one applies.

That was not simply a markup problem.

It was a meaning problem.

We began experimenting.

We developed what we called a **Semantic Data Template**: a way to organize information for machines according to what the information meant rather than merely how it appeared on a page.

We tested representations in YAML and HTML because generative systems already understood those structures.

We began embedding structured datasets within ordinary public web resources.

As those structures became more sophisticated, we separated them according to their semantic functions.

Entity information behaved differently from terminology.

Terminology behaved differently from datasets.

Relationships required their own representation.

Provenance mattered.

Different kinds of machine-facing information required different kinds of containers.

Those became **Fragment Classes**.

The fragments became modular.

The architecture eventually became what we called **Modular Entity Memory**, or **WebMEM®**.

At the time, I was not trying to define a new publishing discipline.

I was trying to communicate more clearly to a second reader.

The first reader was human.

The second was machine.

Then we put the architecture into production.

That is when the more consequential questions began to emerge.

Production use showed that one coherent body of knowledge could participate in questions for which we had never written corresponding answers.

It showed that the number of questions people could ask was vastly larger than the body of underlying knowledge required to support them.

It exposed a problem with the traditional boundaries of web publishing.

A page was not necessarily the natural boundary of knowledge.

Neither was a paragraph.

Neither was a keyword.

Neither was a database row.

The production work repeatedly exposed the need to represent facts together with the context, identity, relationships, and provenance necessary to preserve their meaning when machines retrieved and reused them.

Then we encountered a problem that changed the direction of the research again.

An answer engine could retrieve the correct information and still produce the wrong answer.

Not because the source data was wrong.

Not because the machine necessarily invented a value.

Not because the source lacked authority.

Every retrieved fact could be true.

The unresolved question was which true fact applied.

Retrieval was not resolution.

That distinction exposed another class of publisher knowledge.

If a publisher knows that a fact applies only within a particular geography, time period, configuration, jurisdiction, population, or product variant, that condition is part of what the publisher knows.

If the publisher knows that additional information is required before one of several valid facts can safely be selected, that requirement is also knowledge.

Leaving those conditions implicit requires the consuming system to infer something the publisher already knows.

The deeper the work went, the more the original problem changed.

It was no longer simply:

How do we optimize content for AI?

It became:

What does the machine need to know?

And ultimately:

How should a publisher represent what it knows so that an external machine can use that knowledge correctly?

I did not arrive at these questions as a knowledge engineer.

I do not claim to be one.

I arrived at them as a web publisher attempting to communicate with a new kind of information consumer.

Following that problem downward eventually brought the work into a territory where publishing, data architecture, semantics, information architecture, provenance, and machine reasoning intersect.

I needed a name for the problem we were encountering.

The most accurate one I could find was:

Knowledge Engineering for Answer Engines.

The phrase is not intended as a claim to the invention of knowledge engineering, nor does this study attempt to redefine the established science that carries that name.

It describes the publisher-side problem that emerged from this work:

How should authoritative facts, concepts, relationships, context, provenance, and resolution requirements be organized and published so that external generative systems can use them to construct answers?

This study follows that question from the perspective in which it arose: public web publishing.

It examines what happens when the generated answer is no longer treated as the thing the publisher must necessarily manufacture and the investigation instead moves beneath the answer—to the knowledge from which answers can be constructed.

That means examining the boundaries of knowledge.

Its meaning.

The context that makes a fact applicable.

The identity of the thing being described.

The relationships that make separate facts useful together.

The provenance that establishes authority.

The temporal conditions that establish validity.

And the information required to determine when an answer can—and cannot—be safely resolved.

The work also points toward a different model of publishing.

One in which humans and machines can consume different but aligned representations of the same authoritative reality.

One in which the publisher does not need to anticipate every possible question.

One in which a new information need does not necessarily require another content artifact.

And one in which the answer does not have to exist before the question is asked.

The knowledge does.

This study therefore begins not with an algorithm.

Not with a ranking factor.

Not with a citation tactic.

And not with a claim that the science underlying these problems was invented here.

It begins with a publisher confronting a practical question:

If machines are now interpreting published information before humans encounter the source, how should publishers communicate what they know to both?

Everything that follows came from trying to answer it.

Executive Summary

Generative answer engines are changing the relationship between organizations, published information, and the people who rely on it.

For most of the web's history, that relationship was relatively straightforward. Organizations published documents. Search engines helped people discover them. Human readers visited those documents, interpreted what they found, compared sources when necessary, and reached their own conclusions.

Generative answer systems alter that division of labor.

Increasingly, machines retrieve information from multiple sources, interpret terminology, identify relationships, compare facts, apply context, perform calculations, and synthesize responses before the information consumer encounters the underlying source.

This represents more than a change in search presentation.

It creates a publisher-side information problem.

Organizations are no longer publishing solely for human readers. Their public information increasingly serves as evidence for external machine systems that may retrieve, interpret, recombine, and synthesize that information on behalf of humans.

This study examines what that change requires from publishers.

Research Question

The initial industry response to generative search has understandably concentrated on visibility.

Can an answer engine retrieve the publisher's information?

Will the publisher be cited?

Can existing content be structured so that generative systems can extract and reuse it?

These are legitimate and important questions, commonly addressed through practices described as Answer Engine Optimization (AEO) and Generative Engine Optimization (GEO).

The production work documented in this study exposed a different question:

What knowledge must exist—and how must it be represented—for an answer engine to construct the correct answer in the first place?

Retrievability alone does not determine whether information can be used correctly.

A machine may retrieve an authoritative fact while lacking the context that establishes where or when it applies. It may identify a parent entity while failing to distinguish among variants. It may encounter accurate values without understanding the relationships among them. It may separate an assertion from the provenance that establishes its authority. It may possess several individually valid facts without having enough information to determine which one applies to the question being asked.

The resulting failure is not necessarily a failure of retrieval.

It can be a failure of meaning or applicability.

The publisher-side framework developed in this study is referred to as **Knowledge Engineering for Answer Engines**:

The practice of organizing authoritative facts, concepts, relationships, context, provenance, and resolution rules into machine-consumable knowledge from which generative systems can synthesize answers.

The term is used here to describe the publishing problem encountered through this research. It is not intended as a claim to the invention or redefinition of the established field of knowledge engineering.

The scope of the study is specifically external and publisher-side:

- **The organization possesses authoritative knowledge.**
- **The generative system is external.**
- **The public web sits between them.**

The research question is how the publisher can make its side of that relationship more explicit.

The Question Space and the Knowledge Space

One of the central observations arising from the work is an asymmetry between the number of questions that can be asked about a domain and the amount of underlying knowledge required to support those questions.

A finite collection of facts can support a much larger number of filters, comparisons, counts, rankings, calculations, combinations, and linguistic formulations.

Consider a retailer representing 2,000 cameras through a consistent set of attributes including manufacturer, model, sensor format, resolution, weight, lens mount, stabilization, video capability, and price.

A consumer might ask:

Which full-frame mirrorless cameras under $2,500 weigh less than 700 grams and have in-body stabilization?

The retailer does not need to have anticipated or independently authored that answer.

If the relevant entities, facts, terminology, relationships, and context have already been represented consistently, the question can operate as a projection across knowledge the retailer already possesses.

The knowledge existed.

The answer did not.

The answer was constructed when the question established which portion of the knowledge was relevant and what operation should be performed across it.

This study describes that distinction as the **Question Space / Knowledge Space asymmetry**:

The question space can be enormous while the knowledge space required to support it remains comparatively bounded.

Traditional digital publishing has often responded to expanding information demand by creating additional content artifacts: articles, FAQs, comparisons, location pages, programmatic pages, and increasingly atomic answers intended for machine retrieval.

At sufficient scale, this can produce multiple independently maintained expressions of the same underlying facts.

Generative synthesis makes another publishing model possible:

Represent the underlying knowledge once and allow multiple information needs to project across it.

The governing principle developed from this observation is:

Engineer once. Answer many.

This principle does not eliminate human-oriented content.

Explanation, narrative, analysis, judgment, education, persuasion, interfaces, and decision support remain human communication requirements.

The narrower finding is that not every factual information need necessarily requires another authored content artifact.

From Machine-Readable Data to Machine-Usable Knowledge

Organizations already maintain substantial quantities of machine-readable information in databases, spreadsheets, APIs, product systems, regulatory feeds, research repositories, catalogs, pricing systems, and operational applications.

Machine readability, however, does not by itself establish meaning in the context of a question.

A value of:

$15.26

is machine-readable data.

An assertion that:

$15.26 represents the average Medicare Advantage premium in Putnam County, Ohio, for the 2026 plan year, calculated from a defined eligible population using a specified CMS dataset

contains substantially more information.

The value has acquired identity, geography, time, population, meaning, and provenance.

The production work repeatedly exposed six classes of information required to preserve that meaning during machine use:

- **Facts** — What is asserted?
- **Context** — Where, when, for whom, and under which conditions does the assertion apply?
- **Identity** — What entity does the assertion describe?
- **Relationships** — How do entities, concepts, variants, and facts connect?
- **Provenance** — Where did the assertion originate, and what establishes its authority?
- **Resolution** — Is the available information sufficient to determine which known fact applies?

The objective is not to eliminate machine inference or synthesis.

It is to reduce the need for machines to reconstruct important domain meaning that the publisher already possesses.

Retrieval and Resolution

The distinction between retrieval and resolution became particularly important during production testing involving Medicare plan identifiers.

Retrieval establishes what knowledge is available. Resolution determines which knowledge applies.

A Medicare Plan ID can appear to identify one specific plan while representing a parent entity containing multiple geographic segment variants.

An answer engine may retrieve the correct parent.

It may retrieve the correct children.

It may retrieve the attributes associated with each child accurately.

Yet if geography determines which child applies and the question does not provide geography, the available information remains insufficient for a definitive answer.

All retrieved facts may be valid.

None can yet be established as applicable.

This exposed three separate dimensions of answer correctness:

- **Fact validity** — Is the assertion true?
- **Applicability** — Is the assertion true in the current context?
- **Resolution sufficiency** — Is enough context available to determine applicability?

The research therefore treats resolution requirements themselves as publishable knowledge.

In some cases, the most accurate machine-facing representation is not another factual value but an explicit constraint:

Resolution incomplete. Additional context is required.

The Knowledge Object

The production work also exposed a problem of semantic boundaries.

A web page is a presentation boundary.

A paragraph is a prose boundary.

A keyword represents information demand.

A database row reflects the structure of a source system.

None necessarily corresponds to the natural boundary of the knowledge an external system needs to understand or reason from.

This study uses the term **knowledge object** for:

A coherent and bounded machine-consumable representation of something an answer engine may need to understand or reason from.

A knowledge object may be entity-centric, such as a product and its specifications; context-centric, such as Medicare coverage options within a defined county and plan year; concept-centric, such as a regulated term; or relationship-centric, such as the relationship between a parent plan and its geographic variants.

The knowledge object provides the semantic boundary within which facts, context, identity, relationships, provenance, temporal validity, and resolution semantics can be represented coherently.

It exists independently of any particular question that may later project across it.

A Two-Layer Publishing Architecture

The research further suggests that a modern public web resource increasingly serves two information consumers whose requirements overlap but are not identical.

Human readers require presentation optimized for human cognition and action:

explanation, narrative, navigation, comparison, persuasion, decision support, interaction, and transaction.

Machine consumers benefit from explicit representations of:

identity, facts, context, relationships, canonical concepts, provenance, temporal validity, and resolution requirements.

These do not require different versions of reality.

They can operate as two aligned representations of the same authoritative information:

Human Presentation Layer + Machine Knowledge Layer

Ideally, both originate from a common authoritative knowledge model.

This principle underlies **WebMEM®**, an embedded knowledge-layer architecture developed during the research to publish machine-consumable knowledge within public web resources alongside their human presentation.

WebMEM® is not presented as the definition of Knowledge Engineering for Answer Engines.

It is one implementation architecture developed in response to the publisher-side problem examined in this study.

The broader finding is independent of any particular implementation:

Machine-facing knowledge can be treated as a deliberate publishing layer rather than an incidental byproduct of human-oriented content.

Production Basis

The framework presented in this study emerged from production publishing work involving public Medicare data.

Medicare provided an unusually demanding test environment because its information architecture combines authoritative datasets, geographic variation, temporal variation, hierarchical identifiers, product variants, derived statistics, regulated terminology, provenance requirements, and applicability constraints.

Three production cases were particularly informative.

Medicare Options examined whether one coherent county-level representation of the Medicare coverage environment could support materially different information needs without independently authored answers for each question.

The implementation represented Medicare Advantage, Part D, Medigap, Special Needs Plans, plan counts, plan types, premiums, maximum out-of-pocket observations, carrier presence, geography, plan year, and source provenance within a common contextual boundary.

Observed answer-engine behavior showed the resulting resources participating across multiple factual projections of that underlying knowledge.

Bacon County provided a different observation. The same underlying factual object was independently rendered through two public domains. An obscure factual query caused both implementations to surface as evidence, providing a useful test of factual reuse independent of a heavily optimized generic search query.

Plan-ID exposed the opposite boundary. The answer engine could retrieve the correct plan family and valid child facts while still lacking the contextual discriminator required to determine which child applied.

Together, these cases exposed three distinct questions:

- **What knowledge can be synthesized?**
- **What knowledge can be retrieved?**
- **What knowledge can be correctly resolved?**

The study does not claim access to or knowledge of the internal reasoning, retrieval architecture, ranking systems, or representations used by external answer engines.

The production evidence is evaluated from the publisher side: what was published, what was queried, what appeared, which facts were used, and which sources were surfaced.

Enterprise Implications

The findings extend beyond search and content organizations.

The knowledge required by an answer engine may originate in databases, APIs, Product Information Management systems, CMS platforms, regulatory feeds, research, pricing systems, product catalogs, operational applications, and subject-matter expertise.

Transforming that information into externally usable machine knowledge requires decisions about:

authority;

semantic boundaries;

entity identity;

context;

relationships;

terminology;

provenance;

temporal validity;

applicability;

resolution;

validation;

and governance.

These are not solely content decisions.

They are information-architecture and knowledge-modeling decisions with consequences for external communication.

The resulting enterprise question is therefore broader than:

How visible is our content to AI?

It is:

How much of what our organization knows have we made explicitly usable by machines?

Executive Takeaways

1. Answer engines are becoming direct consumers and interpreters of publicly available organizational information.
2. Machine-readable data is not necessarily machine-usable knowledge; meaning depends on explicit identity, context, relationships, provenance, and applicability.
3. The Question Space is substantially larger than the underlying Knowledge Space, making it possible for one engineered body of knowledge to support many information needs.
4. Retrieval and resolution are separate problems. Retrieving a valid fact does not establish that the fact applies to the current question.
5. Knowledge Objects provide semantic boundaries for representing coherent bodies of machine-consumable knowledge independently of individual questions or pages.
6. Human presentation and machine knowledge can operate as separate but aligned representations of the same authoritative reality.
7. WebMEM® provides one production implementation of an embedded machine knowledge layer; it is an architecture developed through the research, not the definition of the broader framework.
8. Queries can function as tests of the knowledge model, exposing missing facts, relationships, context, terminology, provenance, identity, or resolution rules.
9. Machine-facing knowledge requires continuing validation and governance rather than one-time publication.
10. Knowledge Engineering for Answer Engines is best understood here as a publisher-side framework for a larger enterprise problem: making what an organization knows explicitly usable by external machines.

The models will change.

The interfaces will change.

The dominant answer engines will change.

Organizations will continue to possess authoritative knowledge, and people will continue to ask questions about it.

The enduring problem examined by this study is how effectively publishers can connect the two.

Section 1.
The Answer Engine Changes the Publishing Problem

Generative answer engines are often described as the next evolution of search.

That description is understandable, but it does not fully describe the change observed from the publisher side.

From the user's perspective, the transition can appear incremental. A search box becomes conversational. A ranked list of links gains a generated summary. A traditional results page begins answering questions directly.

From the publisher's perspective, however, the role of the intermediary has changed.

For most of the web's history, search engines principally helped users discover information. Publishers created resources. Search engines indexed and ranked those resources. People selected among them and completed the information task themselves.

Generative answer engines increasingly participate in completing the information task.

That changes what the machine may require from the publisher—and therefore changes the publishing problem itself.

1.1 The Document-Centric Web

The original web was built around documents.

Those documents took many forms: articles, product pages, directories, government publications, research papers, FAQs, reference pages, database reports, and eventually

highly interactive applications. Beneath those differences was a common publishing assumption.

Information was packaged into something a human would retrieve and consume.

The URL became its address.

The page became its container.

And the human reader supplied an enormous amount of interpretation that publishers rarely needed to represent explicitly.

A person visiting a product page can usually determine which specifications describe the product named at the top of the page. A reader examining a table can infer what its rows and columns represent. A consumer reading a county-level insurance page can understand that the statistics on the page apply to the county identified in the heading. A researcher can distinguish a source citation from the assertion it supports.

Humans routinely infer subject, context, hierarchy, relationships, qualifications, and applicability from the organization of a document.

The web therefore became very good at publishing information that contained knowledge without necessarily representing all of that knowledge explicitly.

Search engines evolved within the same architecture.

The basic publishing relationship was:

Publisher → Document → Search Engine → Consumer

The search engine crawled documents, indexed their contents, evaluated their relevance and authority, and returned references to resources likely to satisfy the user's information need.

The information task was generally completed somewhere else.

A person searched.

The search engine located.

The person interpreted.

Search engine optimization developed around that division of labor. Publishers optimized pages, titles, headings, links, site architecture, crawlability, indexability, relevance, authority, and topical coverage because the objective was to make the appropriate document discoverable for the appropriate query.

Keyword research reinforced the same relationship:

Query → Keyword → Content Brief → Page

When new information demand appeared, publishers frequently created additional documents.

A new question became an article.

A new comparison became a comparison page.

A new geography became a location page.

A new long-tail opportunity became another piece of supporting content.

This approach was rational because the page remained the destination. Each new document could become another place where the information task was completed.

The architecture rested on a simple division of responsibility:

The machine finds. The human interprets.

The production observations underlying this study began to suggest that generative answer systems were changing that division.

1.2 When the Answer Becomes the Product

The consequential change introduced by generative answer systems is not simply that search results contain more generated text.

The intermediary can increasingly construct the information product itself.

A person can ask a question in ordinary language. The system may retrieve evidence, interpret terminology, identify entities, combine information from multiple sources, compare facts, recognize relationships, apply context, perform calculations, and synthesize a response.

The user may receive a useful answer without visiting the documents from which the underlying information was obtained.

The emerging information flow can therefore be represented as:

Question → Retrieve Evidence → Interpret Context → Resolve Meaning → Synthesize → Answer

Retrieval has not disappeared.

It remains essential.

But retrieval is no longer necessarily the final objective. It can operate as one stage within a larger process.

This distinction changes the potential role of published information.

A traditional search engine primarily needed to identify a resource likely to contain information useful to a person.

An answer engine can potentially use components of knowledge originating across multiple resources: a product specification from one source, a definition from another, a geographic relationship from another, and a current price or regulatory fact from another.

The resulting answer may exist nowhere verbatim.

The machine can construct an information product from available evidence.

This capability becomes particularly relevant when a question involves comparison, filtering, aggregation, calculation, or relationships.

Suppose a retailer publishes authoritative specifications for every camera it sells. A consumer asks which full-frame mirrorless cameras below a particular price also satisfy requirements for weight, stabilization, and video capability.

The retailer does not necessarily need to have written that comparison.

If the necessary facts are available and represented with sufficient meaning and consistency, a generative system can potentially construct the comparison when the question is asked.

The answer did not need to exist as a document before the information need arose.

This represents a materially different publishing relationship.

The publisher is no longer supplying only destinations for human readers.

It is increasingly supplying evidence from which an intermediary may construct answers.

1.3 The Industry's First Response: AEO and GEO

The publishing and search industries have responded to this transition through practices commonly described as Answer Engine Optimization, Generative Engine Optimization, AI Search Optimization, and related terms.

These practices address an important set of questions:

Can the machine retrieve our information?

Can it interpret the passage?

Will it cite the source?

Can it incorporate the information into a generated answer?

How should content be organized so that generative systems can retrieve and use it effectively?

These are legitimate publishing concerns.

Clear sourcing, structured data, consistent entity signals, retrievable passages, original statistics, useful definitions, FAQs, and well-organized content can all improve the information available to an answer engine.

Nothing observed in this study suggests that these practices are unnecessary.

The production work instead exposed a different limitation.

Much of the AEO and GEO problem can still begin with an anticipated question and an associated answer artifact.

A publisher identifies an information need and determines how to create or optimize information so that a generative system can retrieve, cite, summarize, or reproduce it.

The retrievable unit may become smaller than the traditional page. Instead of optimizing an entire document, a publisher may optimize a paragraph, table, definition, FAQ response, dataset, or other fragment.

That is a meaningful evolution beyond treating every page as an indivisible information unit.

But the underlying publishing model can remain answer-centric:

Anticipate the question. Publish the answer.

At sufficient scale, this risks reproducing the traditional document problem at a finer level of granularity.

More questions produce more atomic answers.

More FAQs.

More fragments.

More comparisons.

More independently maintained expressions of the same underlying facts.

The production work examined in this study suggested another starting point.

1.4 The Deeper Question

Instead of beginning with:

What answer should we publish?

the research began moving toward a different question:

What must the machine know for the answer to be possible?

That change in starting point moves the publishing problem beneath the generated-answer surface.

If a question requires a product price, the machine needs the price.

But depending on the question, it may also need to know which product the price describes, which configuration it applies to, which geography governs availability, when the price is valid, and where the assertion originated.

If a question requires a comparison, the machine may need normalized attributes across multiple entities.

If a question depends on terminology, it may require a stable definition of the concept involved.

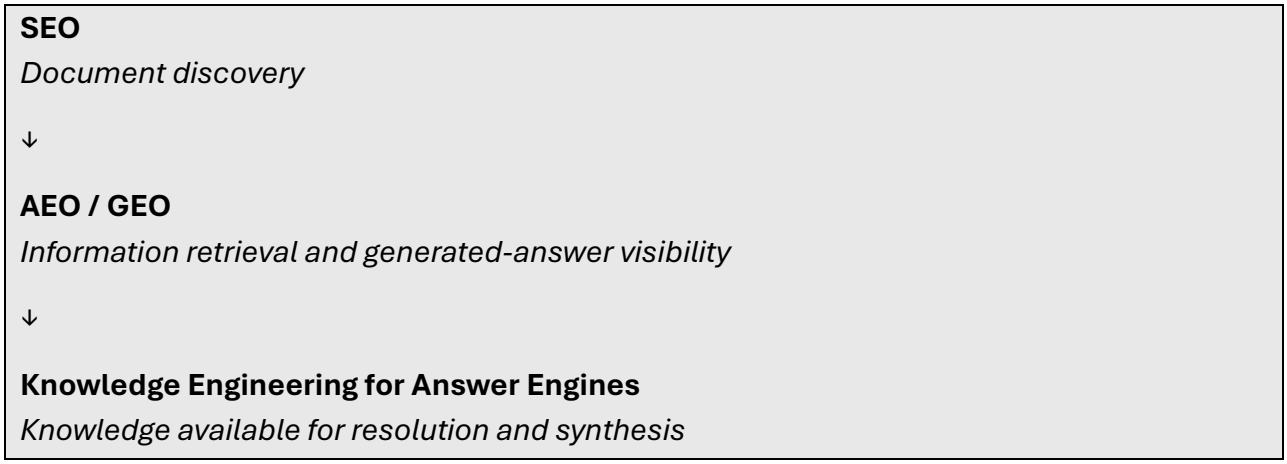
If a product contains variants, the relationship between the parent and its children may determine which facts can safely be used.

And if the available information does not establish which variant applies, the relevant machine-facing knowledge may include something other than another factual value:

The question cannot yet be resolved from the context supplied.

These requirements expose three related layers of the publishing problem.

Figure 2 — Three Layers of the Information Problem



These layers overlap.

SEO remains important because public information generally must be discoverable before it can participate in retrieval.

AEO and GEO address the important problem of making information accessible, interpretable, useful, and attributable within generated-answer environments.

The publisher-side framework examined in this study operates one layer deeper.

It asks what facts, concepts, identities, relationships, context, provenance, temporal conditions, and resolution requirements should be made explicit so that an external generative system has a sufficiently coherent representation of what the publisher knows.

The distinction is between optimizing an information artifact and engineering the knowledge beneath it.

This becomes increasingly consequential as generative systems assume more of the interpretive work historically performed by human readers.

Publishers cannot control how every external model will retrieve, interpret, reason, or synthesize.

They cannot anticipate every question that future users will ask.

The production work examined here instead focuses on what publishers **can** control.

They can establish authoritative facts.

They can make context explicit.

They can identify entities consistently.

They can represent meaningful relationships.

They can preserve provenance.

They can establish temporal validity.

They can communicate conditions governing applicability.

They can identify when additional context is required before a definitive conclusion is justified.

In other words, publishers can engineer the information substrate they make available to external machines.

This study examines the proposition that doing so represents a distinct publisher-side problem from optimizing individual answers for retrieval:

AEO optimizes information for the answer environment. Knowledge Engineering for Answer Engines constructs the substrate from which answers can be made.

Section 2.
The Question Space Is Larger Than the Knowledge Space

The production work examined in this study exposed a recurring asymmetry in digital publishing.

Humans can ask vastly more questions about a domain than there are underlying facts required to answer them.

This appears straightforward once stated, but most digital publishing systems have historically been organized around the opposite side of the relationship. Publishers observe questions, queries, topics, and information needs and create documents intended to satisfy them.

More demand produces more content.

The number of possible questions, however, is not constrained by the number of underlying facts.

A comparatively bounded body of knowledge can support an enormous—and potentially unknowable—question space.

Generative synthesis makes that asymmetry operationally significant.

If a system can construct an answer by filtering, comparing, relating, calculating, or otherwise projecting across existing knowledge, then publishers may not need to create a new information artifact for every new question.

This changes both the economics of publishing and the architecture required to support it.

2.1 Questions Multiply Faster Than Facts

Consider a product described by ten useful attributes.

Each attribute can support a direct factual question.

What does it cost?

How much does it weigh?

What size is it?

What is its capacity?

But information needs do not stop with direct factual retrieval.

People compare.

Which product is lighter?

They filter.

Which products cost less than $500?

They combine conditions.

Which products cost less than $500 and weigh less than two kilograms?

They rank.

Which qualifying product is cheapest?

They calculate.

How much lighter is Product A than Product B?

They ask about relationships.

Which accessories are compatible with this model?

They change context.

Which of these products is available in my location?

They change time.

Which models are currently available?

And they can express the same underlying information need in many different linguistic forms.

A finite set of entities, attributes, relationships, and contextual dimensions can therefore support an enormous number of legitimate questions.

The question space expands combinatorially.

The underlying knowledge does not expand at the same rate.

This distinction matters because the traditional web publishing model frequently treats information demand as a production signal.

A question is discovered.

A content opportunity is identified.

An answer is written.

A resource is published.

That relationship made sense when the publisher needed to create the destination at which the information task would be completed.

Generative systems capable of synthesis introduce another possibility.

If the facts necessary to answer a question already exist in an explicit, coherent, and usable form, the publisher may not need to create another answer artifact.

The question may instead request a new projection across existing knowledge.

The research therefore distinguishes between two related but differently scaled spaces:

Question Space — the potentially enormous set of information needs that can be expressed about a domain.

Knowledge Space — the comparatively bounded set of entities, facts, concepts, relationships, contexts, provenance, and applicability rules required to support those information needs.

The distinction becomes clearer through a simple example.

2.2 The Camera Catalog Example

Consider a photography retailer selling 2,000 cameras.

For every camera, the retailer maintains the same semantic structure:

- manufacturer
- model
- product identity
- camera type
- sensor format
- sensor resolution
- weight
- dimensions
- lens mount
- image stabilization
- video capability
- price

Assume the retailer has created no special content around unusual combinations of these attributes. It simply maintains authoritative, normalized product information.

A consumer asks:

Which full-frame mirrorless cameras under $2,500 weigh less than 700 grams and have in-body stabilization?

That question requires the system to identify cameras where:

type = mirrorless

sensor format = full-frame

price < $2,500

weight < 700 grams

in-body stabilization = true

The retailer did not need to predict that combination.

No keyword researcher needed to discover the query.

No editor needed to commission an article titled *Best Lightweight Full-Frame Mirrorless Cameras Under $2,500 With IBIS*.

No writer needed to create the answer.

No page needed to exist for that precise information need.

Every piece of source knowledge required to construct the answer was already present in the catalog.

The question supplied the projection.

Change the question:

Which qualifying camera has the highest resolution?

No new source knowledge is required.

Resolution becomes another comparison attribute.

Then ask:

Which is the lightest?

Again, no new knowledge.

Which Canon models satisfy the same requirements?

Apply another filter.

Which qualifying cameras use the same lens mount?

Traverse another relationship.

Which model is at least 100 grams lighter than Camera X?

Now the system derives a comparison from existing facts.

None of these information needs necessarily requires another authored answer.

This illustrates the potential value of consistent knowledge structure.

If 2,000 cameras are described independently in prose, a consuming system may first need to extract specifications, reconcile terminology, normalize units, identify entities, and reconstruct a common comparison model before attempting the same operations.

One manufacturer may say “image stabilization.”

Another may say “5-axis sensor-shift stabilization.”

Another may use “IBIS.”

Weight may include a battery in one specification and exclude it in another.

Product configurations may be inconsistently identified.

The machine may therefore need to reconstruct comparability before it can use the facts reliably.

If the publisher has already normalized those cameras into a common semantic structure, that work has been moved upstream.

The publisher has established comparability before the question arrives.

The publishing model consequently changes.

The retailer does not need to possess 50,000 prewritten answers.

It can possess 2,000 consistently represented camera knowledge objects capable of supporting information needs that were not anticipated when the catalog was created.

The knowledge existed.

The answer did not.

The answer could be constructed when the question arrived.

This example is intentionally simple. Real domains introduce additional problems involving terminology, temporal validity, provenance, relationships, variant identity, jurisdiction, eligibility, and other applicability conditions. Those requirements are examined in subsequent sections.

The underlying asymmetry remains the same.

The possible questions multiply much faster than the authoritative knowledge from which they can be answered.

2.3 The Traditional Response: Publish the Question Space

Digital publishing has historically responded to information demand through a familiar strategy:

Discover demand and create content to satisfy it.

The workflow can be represented as:

Query → Keyword → Content Brief → Page

Find another question?

Create another article.

Find a comparison?

Create a comparison page.

Find geographic demand?

Create location pages.

Find a cluster of related questions?

Create supporting content.

Find long-tail variations?

Expand the corpus.

Programmatic publishing industrialized this process by making it possible to generate thousands or millions of pages from structured inputs.

The logic was rational.

Every additional document represented another potential search destination.

But this approach also creates an architectural consequence that can be mistaken for merely a content-quality problem.

The same underlying facts become represented repeatedly.

A product specification may appear on a product page, a category page, several comparison pages, multiple buying guides, FAQs, and programmatic resources.

The organization may possess one authoritative fact conceptually while maintaining dozens or hundreds of operational expressions of it.

Each expression creates another opportunity for:

- inconsistency
- stale information
- contradictory values
- provenance drift
- compliance exposure

- editorial maintenance
- canonical ambiguity

The same architectural pattern can persist when publishing is adapted for generative search.

A publisher may replace large documents with smaller answer-oriented units, but if every identified question produces another independently maintained answer, the underlying relationship remains:

Question → Answer → Published Artifact

At sufficient scale, the organization is still attempting to approximate the Question Space through content.

The difficulty is that the Question Space has no practical boundary.

Users can continuously combine known dimensions in ways the publisher did not anticipate.

Price plus weight.

Weight plus features.

Features plus geography.

Geography plus availability.

Availability plus compatibility.

Compatibility plus price.

Another user can then request a comparison across all of them.

No publishing operation can efficiently prewrite every legitimate permutation of a sufficiently rich domain.

The production work examined in this study suggests that, where the underlying information need can be resolved from explicit knowledge, it increasingly does not need to.

2.4 Knowledge-Space Publishing

Generative synthesis makes another publishing strategy possible.

Instead of attempting to manufacture the Question Space, a publisher can engineer the smaller Knowledge Space beneath it.

Represent the entities.

Represent their facts.

Normalize comparable attributes.

Establish canonical concepts.

Make relationships explicit.

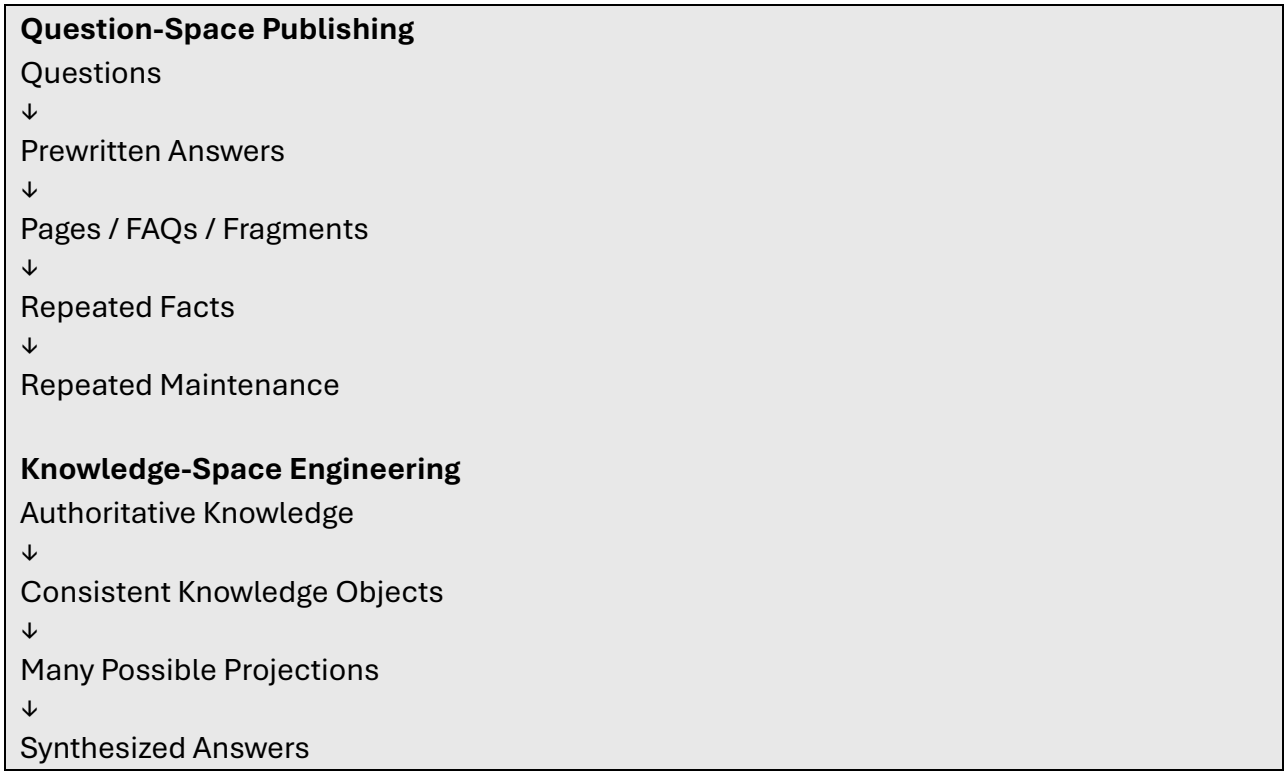
Bind facts to the context in which they apply.

Preserve provenance.

Represent the conditions required for valid interpretation and resolution.

Then allow questions to project across that knowledge.

Figure 3 — Question-Space Publishing vs. Knowledge-Space Engineering



The potential economic difference is substantial.

In the first model, supporting additional information needs generally requires producing additional artifacts.

In the second, adding useful knowledge can expand the range of information needs the existing architecture is capable of supporting.

A single new attribute added consistently across 2,000 camera objects does not merely enable one new answer.

It can participate in combinations with every other relevant attribute already represented.

A new relationship can enable questions across entire classes of entities.

A new contextual dimension can make previously ambiguous information needs resolvable.

Knowledge can therefore produce compounding utility.

This suggests a different way to evaluate publishing efficiency.

The objective is not simply to produce more answers at lower cost.

It is to increase the number and variety of legitimate information needs that can be supported from a maintained body of authoritative knowledge.

This study refers to that principle as:

Engineer once. Answer many.

There is an important boundary to the principle.

Not every information need is a projection across structured knowledge.

People need explanation.

They need analysis.

They need judgment.

They need narrative, persuasion, education, instruction, interpretation, expertise, and decision support.

A photographer may want to understand why one camera system is better suited to wildlife photography than another. Specifications alone may not answer that question. Experience, tradeoffs, interpretation, and judgment may be required.

That is a human communication problem.

A Medicare beneficiary may need to understand the practical differences between Original Medicare and Medicare Advantage.

That requires explanation.

Human-oriented content therefore remains essential.

The proposed change is not that publishers should stop creating content.

It is that publishers should not assume every information need requires another content artifact.

Before creating another answer, the publisher can ask:

Does this information need require new knowledge—or merely a new projection of knowledge already possessed?

If it requires explanation, create the explanation.

If it exposes missing knowledge, engineer the missing knowledge.

If the required knowledge already exists and the question can be safely resolved from it, synthesis can perform the function generative systems are designed to perform.

The Question Space will continue to expand.

The publisher does not need to anticipate all of it.

The publisher needs to understand and deliberately represent the Knowledge Space beneath it.

Section 3.
From Data to Knowledge

If generative answer engines can construct answers from underlying knowledge rather than relying exclusively on prewritten responses, an immediate question follows:

Don't organizations already possess enormous quantities of machine-readable data?

They do.

Governments publish open datasets. Manufacturers maintain product information systems. Retailers operate catalogs and inventory feeds. Financial institutions maintain pricing and product databases. Universities maintain program and admissions data. Healthcare organizations maintain benefits, provider, formulary, eligibility, and enrollment information.

Much of this information is already structured.

CSV is machine-readable.

JSON is machine-readable.

XML is machine-readable.

APIs are machine-readable.

Databases are machine-readable.

Yet the production work examined in this study repeatedly exposed a distinction between making information processable and making its meaning sufficiently explicit for external machine use.

A structured format does not, by itself, establish what a value describes, when it applies, how it relates to other information, where it originated, or whether it can safely be used to answer a particular human question.

The problem is therefore not simply making data machine-readable.

The publisher-side problem examined here is making enough of its meaning explicit.

3.1 Machine-Readable Is Not Machine-Understandable

Machine-readable data solves an important class of problems.

A CSV parser can identify rows and columns.

A JSON parser can identify objects, properties, arrays, and values.

An API can return records matching defined parameters.

A database can retrieve records satisfying a query.

These capabilities allow software to locate, exchange, and manipulate information efficiently.

They do not necessarily establish what the information means in the context of a human question.

Consider a field containing:

15.26

A machine can identify it as a numeric value.

Add a field name:

premium = 15.26

The value now carries additional meaning.

But important questions remain.

Premium for what?

A health insurance plan?

A Medicare Advantage plan?

A prescription drug plan?

One particular product or an average across several products?

Monthly or annual?

For which geography?

During which year?

For which population?

Calculated according to what methodology?

Derived from which source?

A perfectly structured value can remain semantically incomplete.

This distinction becomes particularly important when source data was designed for purposes other than answering human questions.

Administrative databases describe the operational world of the organization that created them.

A government dataset may organize information around contracts, reporting periods, geographic codes, product identifiers, benefit packages, or regulatory categories.

A manufacturer's system may organize products around SKUs, assemblies, revisions, and distribution channels.

A university system may organize information around program codes, campuses, terms, and enrollment categories.

Those structures may be entirely appropriate for their intended purposes.

Human information needs, however, rarely conform precisely to the architecture of the source system.

A Medicare beneficiary does not ask:

Return records matching the applicable Contract ID, Plan ID, Segment ID, county code, and plan-year fields across the relevant CMS datasets.

The beneficiary asks:

What Medicare Advantage options are available in my county?

The difference between those two expressions represents an interpretation gap.

The source systems know how their records are stored.

The publisher understands how those records correspond to the domain.

The answer engine must determine how the available information relates to what the person wants to know.

Making the source accessible through an API does not necessarily eliminate that gap.

An API can make ambiguous information easier to retrieve.

It does not, by itself, resolve the ambiguity.

Transport is not understanding.

Structure is not necessarily meaning.

3.2 Meaning Accumulates Through Context

The distinction becomes clearer by progressively constructing a useful assertion from a single value.

Begin with:

$15.26

The value is precise.

It is machine-readable.

It may even have originated in an authoritative source or authoritative calculation.

By itself, however, it communicates very little.

Add a label:

Premium: $15.26

The semantic content improves.

We now know what type of value is being represented.

Add the subject:

Average Medicare Advantage premium: $15.26

The assertion becomes more specific.

We now know that this is not necessarily the premium of one plan. It is an aggregate describing a Medicare Advantage plan population.

But important applicability conditions remain unknown.

Add geography:

Average Medicare Advantage premium: $15.26
Putnam County, Ohio

The assertion now has a geographic boundary.

Add time:

2026 plan year

The assertion acquires a temporal boundary.

Add the population definition:

Defined eligible Medicare Advantage plan population

The aggregation now has an explicit semantic denominator. The population included in the calculation is defined.

Finally, bind the assertion to authority:

Defined CMS dataset and applicable source version

The original value has now become something substantially richer:

The average Medicare Advantage premium for a defined population of available plans in Putnam County, Ohio, for the 2026 plan year is $15.26, calculated from a specified authoritative CMS dataset and source version.

The number did not change.

Its meaning did.

This progression can be represented simply as:

Data → Context → Knowledge

Context is not decoration around a fact.

It contributes to what the fact means and determines whether that fact is applicable to a particular information need.

Geography can change applicability.

Time can change applicability.

Product configuration can change applicability.

Eligibility can change applicability.

Jurisdiction can change applicability.

Population definition can change applicability.

A value separated from those conditions may remain numerically accurate while becoming operationally incorrect when reused.

A price may be correct in California but incorrectly applied to a consumer in Arizona.

A tax rule may be authentic but belong to another jurisdiction.

A product specification may accurately describe a previous hardware revision.

A Medicare premium may be correct for a different plan year.

A drug cost may be valid for another formulation, plan, pharmacy, or supply period.

In each case, the underlying value can be authentic.

The error occurs because the context governing its applicability has been lost or misapplied.

This distinction is particularly important in generated answers.

An answer does not become correct merely because every number it contains can be traced to an authoritative source.

The facts must belong to the correct context.

Or, stated another way:

The facts must belong to the correct world.

3.3 The Elements of Machine-Usable Knowledge

Context is foundational, but the production work suggests that context alone is insufficient.

As the publishing problem moves from exposing data toward supporting external synthesis, several distinct elements repeatedly become relevant.

Facts

Facts are the substantive assertions.

A camera weighs 658 grams.

Twenty-nine PPO plans are available.

The average premium is $15.26.

A program requires 30 credit hours.

A property contains three bedrooms.

Facts provide the material from which answers can be constructed.

But a fact without sufficient surrounding meaning can be difficult—or unsafe—to apply.

Context

Context establishes the conditions under which a fact carries its intended meaning.

Depending on the domain, context may include:

- geography
- time
- jurisdiction
- population
- product configuration
- eligibility
- service area
- measurement conditions
- regulatory period

Context answers:

Where, when, for whom, and under which conditions is this assertion true?

The more conditional the domain, the more consequential explicit context becomes.

Identity

Every fact describes something.

That appears trivial until names, variants, aliases, parent-child structures, and multiple source systems are involved.

A display name is not necessarily sufficient identity.

A product can have a marketing name, model number, SKU, regulatory identifier, and several configurations.

A company can have a legal name, brand name, former name, subsidiaries, and abbreviations.

A Medicare plan can have a Contract ID, Plan ID, and Segment ID.

Stable identity establishes what the assertion actually describes.

Without sufficient identity, a fact can be correctly retrieved and attached to the wrong entity or variant.

Relationships

Knowledge is not simply a collection of entities and attributes.

Meaning also exists between things.

A plan is offered by an organization.

A plan segment is a variant of a parent plan.

A PPO is a type of Medicare Advantage plan.

A camera uses a particular lens mount.

A county is located within a state.

These relationships may be expressed in prose, but the relationship itself is structural knowledge:

Plan → offeredBy → Organization

Segment → variantOf → Parent Plan

Camera → usesMount → Lens Mount

Making relationships explicit allows consuming systems to work across entities and concepts without requiring every relevant connection to have been independently authored as prose.

Provenance

Assertions also have origins.

Where did the fact come from?

Which dataset?

Which version?

Which measurement period?

Which manufacturer specification?

Was the value published directly, normalized from a source, calculated, aggregated, or inferred?

Traditional publishing often presents provenance as something attached to a claim through a citation, source note, or footnote.

For machine-facing knowledge, the research suggests that provenance can be treated as part of the assertion's usable meaning.

It establishes where the assertion originated, why it may deserve authority, and what lineage produced it.

This becomes particularly important for derived knowledge.

A government agency may publish thousands of individual plan records without ever publishing the statement:

Average premium in Putnam County = $15.26.

A publisher may calculate that value from authoritative records.

The resulting assertion can remain useful and traceable, but its derivation matters.

A consuming system should ideally be able to distinguish a source assertion from an assertion calculated or aggregated from source facts.

Resolution

Finally, some knowledge cannot safely be applied merely because it has been retrieved.

A product identifier may identify a family rather than a specific configuration.

A Medicare Plan ID may identify multiple geographic segments.

A telecom provider may serve a ZIP code without serving every address within it.

A replacement part may fit one engine configuration but not another.

A drug's coverage may depend on formulation, dosage, plan, pharmacy, and supply period.

In these cases, the publisher may possess all relevant candidate facts while also knowing that additional context is required before one can be selected as applicable.

That requirement is itself knowledge.

Resolution asks:

What must be known before this assertion can safely be applied?

Section 5 examines this problem in greater depth because production testing exposed an important distinction between retrieving valid information and producing an applicable answer.

At this stage, the important finding is narrower:

Machine-usable knowledge may need to represent not only what is true, but the conditions under which a truth can be safely applied.

Figure 4 — From Data to Knowledge

+ CONTEXT

Where, when, for whom, under what conditions?

+ IDENTITY

What does this describe?

+ RELATIONSHIPS

How does it connect to other entities and concepts?

+ PROVENANCE

Where did it come from, and what establishes its authority?

+ RESOLUTION

What must be known before it can safely be applied?

↓

MACHINE-USABLE KNOWLEDGE

The conceptual transformation is straightforward.

The implementation can be difficult.

Organizations already possess substantial quantities of authoritative data. What may remain implicit is the meaning that their own people, applications, business rules, source systems, and institutional practices routinely supply around that data.

People inside the organization know that one identifier represents a parent and another represents a child.

They know that a price applies only in a particular market.

They know that two differently named fields represent the same concept.

They know which source takes precedence.

They know that a value expires at the end of the year.

They know that geography must be supplied before one of several valid records can be selected.

That knowledge may already exist operationally.

It simply may not exist in a representation that an external answer engine can reliably encounter and use.

This leads to a governing principle for the publisher-side framework examined in this study:

Do not make the machine infer what the publisher already knows.

The objective is not to eliminate inference.

Generative systems are useful precisely because they can interpret incomplete, heterogeneous, and previously uncombined information.

The objective is to reduce **unnecessary inference about facts, identity, context, relationships, authority, and applicability that the publisher is already in a position to make explicit.**

Once those elements are brought together, the publisher is no longer merely exposing data.

It is creating a machine-consumable representation of something the organization knows.

The next question is how to determine what belongs together.

That requires a semantic publishing boundary.

Section 4.
The Knowledge Object

If the publisher's role is no longer necessarily to prewrite every possible answer, a practical question follows:

What should the publisher create instead?

Section 3 established that data becomes substantially more useful for external machine consumption when facts are bound to identity, context, relationships, provenance, and the conditions governing their applicability.

But those elements cannot simply be scattered across a website or placed into an undifferentiated structured-data container.

Knowledge requires boundaries.

Some facts naturally belong together. Others do not. Some definitions should be shared across thousands of resources. Some relationships connect otherwise independent entities. Some facts are meaningful only within a particular geographic, temporal, product, or regulatory context.

The web has long had a convenient unit for managing information: the page.

The production work examined in this study exposed the need for an equivalent semantic unit.

This study refers to that unit as the **knowledge object**.

4.1 Why a Semantic Unit Is Needed

The page is an extraordinarily useful publishing abstraction.

It gives publishers something bounded that can be created, named, linked, indexed, updated, measured, governed, and retrieved.

But the page is fundamentally a **presentation boundary**.

A single page can contain several unrelated bodies of knowledge. It can contain one coherent body of knowledge surrounded by explanation and navigation. It can contain only part of a larger body of knowledge. And knowledge represented across several different pages may actually describe one coherent thing.

The URL therefore does not necessarily define the natural boundary of the knowledge beneath the presentation.

Neither does the paragraph.

A paragraph exists primarily because prose requires structure. One paragraph can contain several facts. One fact can depend on information established several paragraphs earlier. Relationships may be implied across sentences. Context may come from a heading or table label located elsewhere on the page.

The paragraph is a **prose boundary**, not necessarily a knowledge boundary.

The keyword presents a different problem.

Keywords describe expressions of information demand.

They tell us something about what people ask and how they ask it.

They do not necessarily tell us how the underlying reality should be represented.

Twenty different queries may depend on the same underlying knowledge. One query may require knowledge from several different entities or domains.

The keyword is a **demand signal**, not a semantic unit.

Even the database row is insufficient as a general-purpose knowledge boundary.

A database is structured, but its boundaries reflect the operational architecture of the system that created it.

One row may represent a product.

Another may represent a product-location relationship.

Another may represent one benefit.

Another may represent one monthly observation.

Another may represent a geographic variant.

A human information need rarely depends on how the source database was normalized.

Ten tables may contribute to one coherent body of knowledge.

One database record may contribute to several.

The database row is a **source-system boundary**, not necessarily a semantic boundary.

The production work therefore exposed a different design question:

What belongs together because it describes the same meaningful thing or state?

That question moves the publishing boundary away from presentation, demand, and source-system structure and toward meaning.

It requires translating the structure of the underlying reality into the structure of publishing.

4.2 The Knowledge Object

For purposes of this study:

A knowledge object is a coherent and bounded machine-consumable representation of something an answer engine may need to understand or reason from.

Each part of the definition establishes an architectural requirement.

Coherent means that the information belongs together semantically. Facts are not grouped merely because they happen to appear on the same page or originate in the same database.

Bounded means that the object represents something identifiable rather than attempting to contain everything the publisher knows about an entire domain.

Machine-consumable means that important meaning is represented explicitly enough that a consuming system does not have to reconstruct all of it from human prose, page structure, or source-system conventions.

And **something an answer engine may need to understand or reason from** deliberately separates the object from any one particular question.

A knowledge object is not simply a structured answer.

If an object exists only to reproduce one anticipated answer in another format, the publisher has recreated Question-Space publishing inside the machine layer.

The object instead represents a meaningful piece of reality from which multiple legitimate answers may potentially be constructed.

Consider a camera.

A useful camera knowledge object might contain:

- manufacturer
- model
- stable product identifier
- camera type
- sensor format
- resolution
- weight
- dimensions
- lens mount
- stabilization
- video capability
- price
- applicable relationships
- provenance

Why do those facts belong together?

Not because they came from the same spreadsheet.

Not because they appeared on the same product page.

They belong together because they describe one identifiable product entity.

The camera provides the semantic center.

Now consider something substantially different:

2026 Medicare coverage options in Mohave County, Arizona.

No single product provides the center.

The meaningful thing being represented is a **contextual state**: the Medicare coverage environment available within a defined geography and period.

That object might contain information about Medicare Advantage, Part D, Medigap, Special Needs Plans, plan counts, plan types, premiums, carrier presence, and other local facts.

Those facts may originate in different datasets.

They may describe different branches of Medicare.

But collectively they describe one coherent state:

the Medicare option space available in Mohave County during the 2026 plan year.

Meaning—not the source file—defines the boundary.

This distinction became fundamental to the architecture developed through the production work.

4.3 Knowledge Objects Can Represent Different Kinds of Things

The research did not support forcing every useful representation into a single entity-centric model.

Different kinds of knowledge can have different natural semantic centers.

Entity-Centric Knowledge Objects

Some objects organize naturally around a stable entity.

A camera is an obvious example.

So is:

- a vehicle

- a drug
- a Medicare plan
- a university
- a property
- a company

The entity supplies the semantic center.

Facts and relationships belong within or around the object because they describe that identifiable thing.

Context-Centric Knowledge Objects

Other knowledge is organized around a defined state or environment rather than a conventional entity.

2026 Medicare coverage options in Mohave County, Arizona is one example.

A residential electricity-rate environment for customers in a defined utility territory during a particular rate period could be another.

The context itself establishes the semantic boundary.

This is significant because many human information needs concern not one thing, but **what is true within a particular set of conditions**.

Geography, time, eligibility, jurisdiction, configuration, or population can therefore participate in the identity of a knowledge object rather than merely being attached as descriptive metadata afterward.

Concept-Centric Knowledge Objects

Some objects exist primarily to establish meaning.

Consider:

Maximum Out-of-Pocket Limit

A concept object might contain:

- canonical name

- abbreviation
- definition
- domain
- synonyms
- related concepts
- broader or narrower concepts
- authoritative source
- important qualifications

The purpose of this object differs from that of a county premium or camera specification.

It establishes canonical meaning that other objects can reference.

The definition does not need to be reconstructed independently every time MOOP appears.

Meaning can be established once and referenced repeatedly.

Relationship-Centric Knowledge Objects

Some knowledge exists primarily in the connection between things.

For example:

Plan Segment A → variantOf → Parent Plan B

or:

Camera X → usesMount → RF

or:

Plan Y → offeredBy → Organization Z

The relationship itself is an assertion.

It carries meaning independently of whether that relationship has been expressed as a sentence for a human reader.

Relationships can also support information needs that were never independently modeled as answers.

If the publisher has represented:

Plan A → typeOf → PPO

Plan A → offeredBy → Carrier X

Plan A → availableIn → Putnam County

then the knowledge required to answer:

Does Carrier X offer PPO plans in Putnam County?

already exists.

The answer may not.

This is consistent with the Question Space / Knowledge Space distinction examined in Section 2.

4.4 The Semantic Contract

A useful way to interpret the knowledge object is as a **semantic contract** between the publisher and a consuming system.

The object effectively establishes:

This is what I represent.

Its identity and type establish the subject.

These are the facts associated with it.

Its attributes establish what the publisher knows about that subject.

This is the context in which those facts apply.

Geography, time, configuration, population, eligibility, jurisdiction, or other conditions establish applicability.

These are the entities and concepts to which it relates.

Relationships place the object within a larger knowledge environment.

This is where the assertions came from.

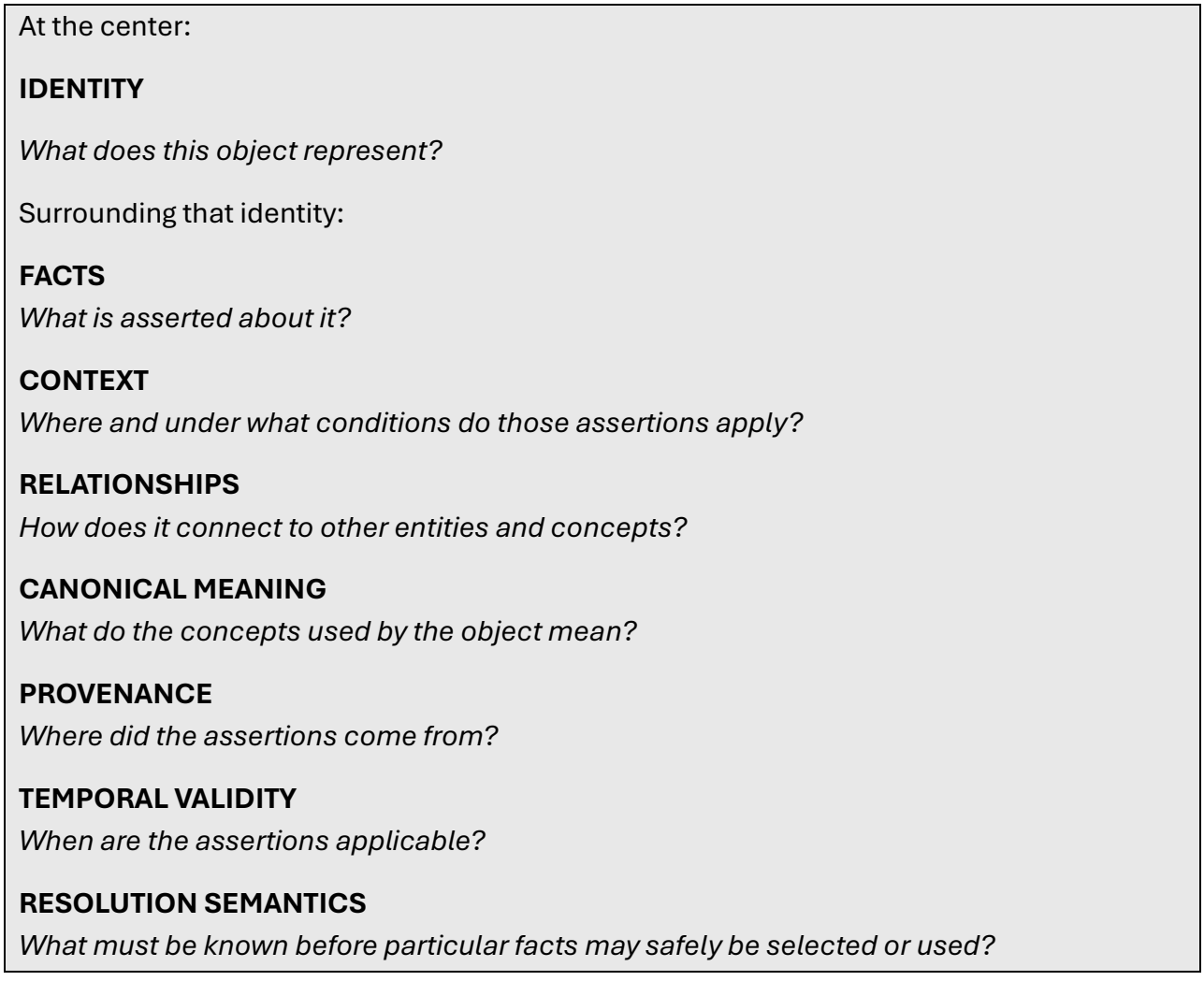
Provenance establishes source, version, authority, and lineage.

These are the limits on how the assertions may be interpreted.

Temporal validity, variant structure, applicability conditions, and resolution requirements establish boundaries on valid use.

The more explicit that semantic contract becomes, the less domain meaning an external system must reconstruct from nearby prose, page layout, database conventions, or assumptions.

Figure 5 — Anatomy of a Knowledge Object



Together, these elements form the object's semantic contract.

Not every knowledge object requires every element in equal depth.

A canonical terminology object may have little geographic context.

A product object may require substantial configuration identity.

A regulatory object may depend heavily on jurisdiction and effective dates.

A resolver object may be dominated by applicability conditions.

The architecture should follow the meaning of the knowledge rather than force every object into an identical container.

The same principle applies to granularity.

A knowledge object must be bounded.

It does not need to be tiny.

If every individual value becomes its own object, the consuming system may be forced to reconstruct the context that made those values meaningful in the first place.

At the opposite extreme, placing everything an organization knows about an entire domain into one enormous object destroys useful semantic boundaries.

The appropriate level lies between meaningless atomization and indiscriminate aggregation.

The production work suggests a practical test:

What piece of reality does this object describe?

If that question cannot be answered clearly, the semantic boundary likely requires additional work.

Once multiple objects of the same type share a common semantic contract, another useful property emerges:

They become inherently comparable.

Two thousand camera objects represented consistently can support filtering, ranking, comparison, and derivation across the catalog.

Thousands of geographic option-space objects represented consistently can support similar operations across locations.

Common structure creates computational leverage.

But the defining characteristic of the knowledge object may be what it is **not**.

It is not the page.

It is not the database row.

It is not the query.

It is not the interface.

And it is not the answer.

A camera object does not exist solely to answer:

How much does this camera weigh?

It exists to represent the camera.

Weight is one fact within that representation.

The same object may later support questions about price, resolution, stabilization, compatibility, video capability, or comparisons with products the publisher never anticipated anyone would make.

The object exists before those questions arrive.

That is the architectural distinction examined in this study:

The knowledge object is upstream of the answer.

The page presents.

The question requests.

The answer engine synthesizes.

The knowledge object represents what is known.

Once that distinction is established, another problem becomes visible.

An answer engine may retrieve the correct knowledge object, encounter valid facts and relationships, and still lack sufficient information to determine which fact applies to the question in front of it.

That is where retrieval ends.

And resolution begins.

Section 5.
Retrieval Is Not Resolution

For much of the history of web search, retrieval was the central problem.

A user expressed an information need. The search system attempted to identify the documents most relevant to that need. If the appropriate resource appeared, the system had largely performed its function.

Generative answer engines face an additional problem.

Finding relevant information may be only the beginning.

Before a system can construct a correct answer, it may also need to determine which entity the user means, which version or variant of that entity applies, which geographic or temporal context governs the question, and which of several individually valid facts is applicable.

The production work examined in this study exposed a distinction that became central to the publisher-side framework:

Retrieval establishes what knowledge is available. Resolution determines which knowledge applies.

The difference can be subtle.

It can also separate an answer that is completely supported by authoritative facts from an answer that is actually correct.

5.1 Candidate Knowledge vs. Applicable Knowledge

Retrieval asks:

What information appears relevant?

Resolution asks:

Which entity, variant, context, or fact actually applies?

Those are different operations.

Suppose someone asks about the price of a particular product.

A retrieval system may locate the correct product family and identify several prices associated with it.

That information is relevant.

But the existence of several prices introduces another question.

Why are there several?

Perhaps the product has multiple configurations.

Perhaps price varies by geography.

Perhaps some prices are historical.

Perhaps one applies to retail customers and another to commercial customers.

Perhaps several child products share the same parent identifier.

Retrieval has produced a set of plausible facts.

It has not necessarily produced the answer.

This study refers to those plausible facts as **candidate knowledge**.

Resolution is the process of determining which candidate knowledge becomes **applicable knowledge** for the question being asked.

The same distinction can arise anywhere facts depend on context.

A replacement part may be relevant to a vehicle model but fit only one engine configuration.

An internet provider may operate within a ZIP code but serve only some addresses.

A financial product may exist nationally while its rate varies by state, term, amount, or eligibility.

A drug may be covered by a health plan while its cost depends on formulation, dosage, pharmacy, and supply period.

In each example, retrieval can place the system in the correct informational neighborhood without establishing the applicable fact.

Relevance narrows the candidates.

Resolution determines applicability.

5.2 The Plan-ID Problem

Medicare provided a particularly useful production example because ambiguity can exist inside something that appears unambiguous: an identifier.

Consider:

H5216-043

The identifier appears precise.

It contains a Contract ID and Plan ID.

Unlike a vague phrase such as “that Medicare plan I was looking at,” the identifier appears to have resolved the entity.

Production work with Medicare plan data demonstrated that this assumption can be incomplete.

A Medicare plan can also contain a **Segment ID**.

Conceptually:

H5216-043

↓

H5216-043-1
H5216-043-2
H5216-043-3

The parent identifier identifies the plan at one level.

The Segment ID identifies a child variant at a deeper level.

The relationship can be represented explicitly:

H5216-043-1 → variantOf → H5216-043

H5216-043-2 → variantOf → H5216-043

H5216-043-3 → variantOf → H5216-043

This led to the useful concept of **resolution depth**.

An identifier can be sufficiently specific for one question and insufficiently specific for another.

A Contract ID identifies less than a Contract ID plus Plan ID.

A Contract ID plus Plan ID may identify less than a Contract ID plus Plan ID plus Segment ID.

The resolution depth required depends on the information being requested.

The parent identifier may be sufficient to answer:

Which organization administers this plan family?

But it may be insufficient to answer:

What is the premium for this plan?

The reason is structural.

Child variants can contain different values.

Knowing the parent does not necessarily establish which child is applicable.

5.3 All the Facts Can Be True and the Answer Still Wrong

This produces one of the more consequential findings from the Medicare production work.

Suppose the available knowledge contains:

Variant A → Premium = $0

Variant B → Premium = $18

Variant C → Premium = $27

Assume each value originates in the appropriate authoritative source.

There is no fabricated number.

There is no stale record.

There is no failure to retrieve the relevant family of information.

All three facts are valid.

Now suppose geography determines which segment applies, and the user asks only:

What is the premium for H5216-043?

The available information does not justify selecting $0.

It does not justify selecting $18.

It does not justify selecting $27.

A generative system might summarize the candidate values:

Premiums range from $0 to $27.

That statement is compatible with the retrieved facts.

It may even appear useful.

But if exactly one segment applies to the person asking the question, the range has not resolved the information need.

The user does not have a premium ranging from $0 to $27.

The user has one applicable segment with one applicable premium.

The missing information is not another premium value.

It is the context required to determine which premium applies.

This exposes an important distinction:

Factual consistency is not the same as resolution accuracy.

An answer can contain no invented information and still be incorrect for the question being asked.

This is also why hallucination is an incomplete model for answer-engine failure.

A generated response does not always fail because the system fabricated information.

It may fail because valid information was selected, combined, generalized, or applied incorrectly.

The failure can occur at the level of applicability.

5.4 Three Different Questions

Once applicability enters the architecture, factual correctness separates into at least three questions.

Fact Validity

Is the assertion true?

Does authoritative evidence establish that:

Variant B premium = $18

If so, the fact itself is valid.

Applicability

Is the assertion true in the current context?

If the user's geography maps to Variant B, the $18 premium is applicable.

If the user's geography maps to Variant A, the $18 premium remains a valid fact, but it is not the applicable fact for this question.

Fact validity and applicability are therefore different properties.

Resolution Sufficiency

Do we possess enough context to determine applicability?

If the question identifies only the parent plan and geography is required to identify the applicable segment, resolution is incomplete.

The publisher's knowledge may be complete.

Every child may be represented.

Every premium may be accurate.

Every relationship may be explicit.

Yet the available question context may still be insufficient to select one definitive premium.

The missing information is not necessarily missing from the publisher's knowledge.

It is missing from the context supplied with the information request.

This is a fundamentally different class of problem.

Adding more facts does not necessarily solve it.

Adding more pages does not solve it.

Making existing passages easier to retrieve does not solve it.

The system requires another discriminator.

The resulting model can be represented as:

Fact validity
Is the knowledge true?

↓

Applicability
Does this knowledge govern this case?

↓

Resolution sufficiency
Do we know enough to determine that?

A definitive answer should follow only when the available information supports the required level of resolution.

5.5 Resolution Rules as Knowledge

This finding led to a less conventional proposition:

Publishers can represent not only facts about the world, but also **conditions governing when those facts may safely be applied**.

Knowledge is often represented as an assertion:

X has value Y.

But another useful form is:

X has value Y when condition Z is satisfied.

Or:

Do not select a child value until geography is known.

Or:

Do not treat a parent entity and its variants as interchangeable for this attribute.

Or:

Do not apply this rate outside the specified jurisdiction.

Or:

Do not use this value after its effective period expires.

Publishers frequently already possess these rules.

They may exist inside application code.

They may be implemented through database joins.

They may appear in technical documentation.

They may exist as business rules.

They may be understood by subject-matter experts.

The organization's own systems may already depend on them to produce correct results.

From the publisher-side perspective examined in this study, that raises an important question:

If the publisher knows the rule required to interpret the knowledge correctly, why leave that rule implicit for an external machine to reconstruct?

This is the basis for treating **resolution rules as knowledge**.

Conceptually, a knowledge representation might communicate:

Entity: H5216-043

Resolution state: Parent plan identified

Variant state: Multiple segment variants exist

Affected attributes: Segment-specific values differ

Required context: Geography

Constraint: Do not present segment-specific attributes as definitive parent-level facts

In the WebMEM® architecture examined later in this study, this condition can be represented through a resolution requirement, conceptually expressed as:

resolution_required

The purpose of resolution_required is not to communicate:

We don't have the data.

The publisher may possess all relevant candidate data.

It communicates something different:

The available knowledge is known, but additional context is required before one candidate can be established as applicable.

This allows the knowledge representation to express not merely what is known, but the **boundary of valid assertion**.

In a conversational environment, that boundary can potentially become actionable.

Instead of collapsing several premiums into a range, the system can request:

Which county do you live in?

Instead of guessing whether an automotive part fits:

Which engine does your vehicle have?

Instead of broadly asserting telecom availability:

What is the service address?

Instead of guessing drug coverage:

Which formulation and plan are you asking about?

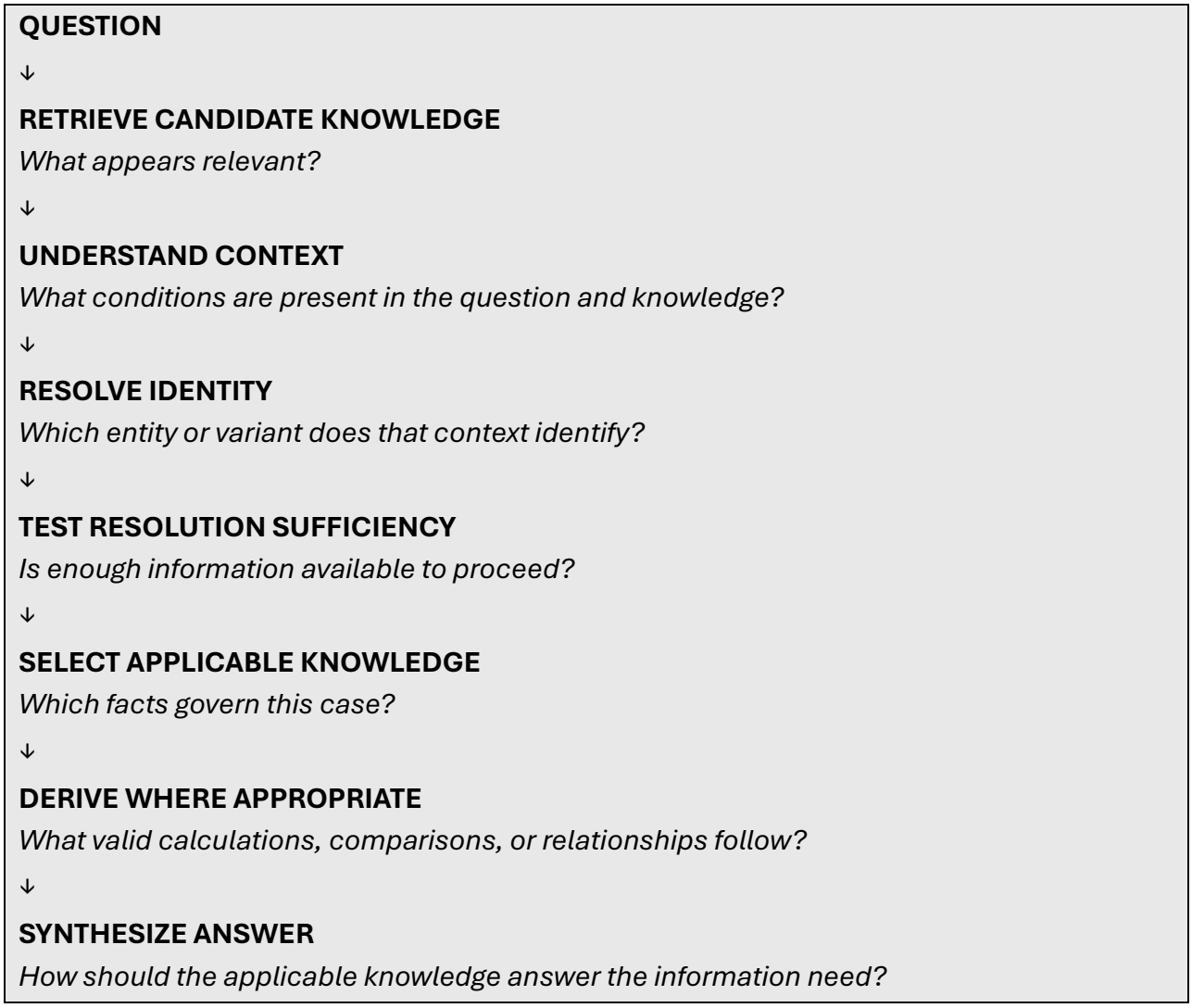
The system can move from ambiguity toward resolution.

This changes the potential function of the publisher's machine-facing knowledge.

The publisher is not merely supplying candidate evidence.

It can also expose the conditions required to determine which evidence applies.

Figure 6 — The Mature Answer Pipeline



This model is substantially more complete than:

Retrieve passage → Generate response

It also illustrates why publisher-side architecture eventually encounters problems that cannot be addressed solely by optimizing passages for retrieval.

The publisher does not control every stage of this pipeline.

External answer engines determine how they retrieve, interpret, clarify, calculate, and synthesize.

Nor does publishing a resolution requirement guarantee that an external system will honor it.

What the publisher can control is the information substrate made available to those systems.

The publisher can make identity explicit.

It can expose variant structure.

It can bind facts to context.

It can preserve temporal applicability.

It can establish relationships.

It can communicate which discriminators matter.

And it can represent when the available context is insufficient to justify a definitive conclusion.

These measures do not guarantee correct reasoning by an external system.

They reduce the amount of important domain logic the system must reconstruct or infer independently.

This is the point at which the publisher-side framework examined in this study moves materially beyond conventional retrieval optimization.

The objective is no longer merely:

Make the right facts retrievable.

It becomes:

Make the right facts retrievable, interpretable, and resolvable.

Because the most relevant fact is not necessarily the applicable fact.

And a definitive answer is not always the correct behavior.

Sometimes the more accurate response is to recognize that resolution is incomplete.

Retrieval establishes what knowledge is available. Resolution determines which knowledge applies.

Section 6.
The Two-Layer Web

The architecture developed through the preceding sections leads to a relatively simple proposition.

A modern public web resource increasingly serves two information consumers.

One is human.

The other is machine.

For most of the web's history, publishers did not need to make this distinction explicit. Pages were designed primarily for people. Search engines crawled those pages, indexed them, extracted signals from them, and attempted to understand them well enough to determine when they should be retrieved.

The information task, however, was ultimately completed by a human.

Generative answer engines alter that relationship.

The machine is no longer limited to determining whether a page might help a person answer a question.

It can increasingly attempt to determine **what the publisher knows** so that the information can participate in an answer constructed for the user.

The production work examined in this study suggests that these two consumers have overlapping but different representational requirements.

Humans need presentation.

Machines benefit from explicit knowledge.

Both should describe the same underlying reality.

They do not necessarily need to represent that reality in the same way.

This study refers to the resulting publisher-side architecture as:

The Two-Layer Web.

6.1 Two Consumers

Consider what happens when a person visits a product page.

The person may want to understand what the product does, how it differs from alternatives, whether it fits their needs, what other people think about it, what it costs, and what action to take next.

The page may therefore contain:

- photographs
- explanatory copy
- headings
- specification tables
- comparisons
- reviews
- buying guidance
- navigation
- financing information
- shipping information
- calls to action

These elements exist because humans consume information through an experience.

We scan.

We read.

We compare.

We interpret visual hierarchy.

We follow narrative.

We respond to emphasis.

We make judgments.

We take actions.

Now consider an answer engine encountering information about the same product.

Its information requirements may be different.

Depending on the question, the system may need to determine:

- What product is this?
- Who manufactures it?
- Which product category does it belong to?
- What are its specifications?
- Which configuration do those specifications describe?
- What units are being used?
- What other entities does it relate to?
- Which products are meaningfully comparable?
- Where did these facts come from?
- When are they valid?
- Does the question identify the product precisely enough for the requested attribute?

The human and the machine are working with the same underlying reality.

But they are performing different information tasks.

Their requirements overlap.

They are not identical.

The production work therefore suggests that attempting to make one representation simultaneously optimize every requirement for both consumers may be unnecessary.

A publisher can instead maintain aligned representations designed around the needs of each.

6.2 The Human Presentation Layer

The traditional web page remains highly effective at serving humans.

Nothing observed in this study suggests that should change.

The **Human Presentation Layer** is designed for:

- explanation
- narrative
- navigation
- comparison
- persuasion
- decision support
- interaction
- transaction

Its architecture reflects human cognition and behavior.

Headings communicate hierarchy.

Paragraphs explain.

Tables help people compare.

Images communicate things prose cannot.

Navigation establishes orientation.

Examples create understanding.

Calls to action help people complete tasks.

Narrative connects facts into meaning.

Design communicates importance.

These functions do not become obsolete because machines are also consuming information from the web.

Human communication remains a distinct publishing problem that should continue to be optimized for human needs.

A first-time Medicare beneficiary may not simply need a list of factual differences between Original Medicare and Medicare Advantage. The beneficiary may need an explanation that makes the practical consequences of choosing one path over another understandable.

A photographer comparing two camera systems may need more than specifications. The decision may require expert interpretation of tradeoffs involving lenses, ergonomics, autofocus, weight, workflow, and long-term system investment.

A university applicant may need more than tuition, prerequisites, and program duration. The applicant may need help understanding whether the program is appropriate for particular goals.

Those are human information experiences.

They require explanation, interpretation, judgment, and presentation.

The Human Presentation Layer should remain free to serve those needs well.

6.3 The Machine Knowledge Layer

The machine-facing information problem is different.

A consuming system does not necessarily require persuasive sequencing.

It does not inherently require visual hierarchy.

It does not require a transition paragraph between two factual sections merely because that transition makes the document easier for a person to read.

The publisher-side objective is to make the knowledge represented by the resource sufficiently explicit for machine use.

The **Machine Knowledge Layer** examined in this study is therefore organized around different requirements.

Identity

What does this resource or knowledge object represent?

Which entity, concept, product, location, configuration, or contextual state has been identified?

Explicit Facts

What does the publisher assert about it?

Values, attributes, classifications, availability, measurements, and other factual assertions need explicit subjects rather than relying solely on their position within a human document to establish meaning.

Context

Where, when, for whom, and under which conditions do those facts apply?

Geography, time, jurisdiction, configuration, eligibility, and population can determine whether an otherwise valid fact is applicable.

Relationships

How does one thing connect to another?

A product may be manufactured by a company.

A plan may be offered by a carrier.

A segment may be a variant of a parent plan.

A camera may use a particular lens mount.

The relationships themselves carry knowledge.

Canonical Concepts

What do recurring terms mean?

A contextual object should not need to redefine the same domain concept every time it appears.

Canonical meaning can be established once and referenced where needed.

Provenance

Where did the assertion originate, and what establishes its authority?

Which authoritative source, dataset, version, specification, methodology, or derivation produced it?

Temporal Validity

When is the knowledge applicable?

Publication date, retrieval date, effective date, measurement period, plan year, and source version describe different temporal properties and should not be treated as interchangeable.

Resolution Requirements

Does the available identity and context justify a definitive conclusion?

If another discriminator is required, the machine-facing representation can make that requirement explicit rather than leaving the consuming system to infer it.

The two layers therefore perform different functions.

The Human Presentation Layer **explains and communicates**.

The Machine Knowledge Layer **makes meaning explicit**.

6.4 Same Reality, Different Representation

The two-layer model raises an immediate and important concern.

If humans receive one representation and machines receive another, does this amount to cloaking?

The architecture examined in this study requires the opposite principle:

Alignment.

The Human Presentation Layer and Machine Knowledge Layer should represent the same underlying reality.

The distinction is not intended to create different factual worlds for different consumers.

If a camera weighs 658 grams, the human specification table should not say 658 grams while the machine representation says 642 grams.

If 29 PPO plans are available within a defined county and plan year, the machine representation should not assert 31.

If a fact requires geographic resolution, the human application should not recognize that requirement while the machine representation presents the same fact as universally applicable.

The difference is not truth.

The difference is **representation**.

A human-facing resource might state:

There are 29 PPO Medicare Advantage plans available in Putnam County for 2026.

A machine-facing representation might express the same underlying assertion as distinct semantic components:

Geography: Putnam County, Ohio
Plan year: 2026
Product domain: Medicare Advantage
Plan type: PPO
Count: 29
Source: defined CMS dataset/version
Relationship: PPO → typeOf → Medicare Advantage

Same underlying reality.

Different representation.

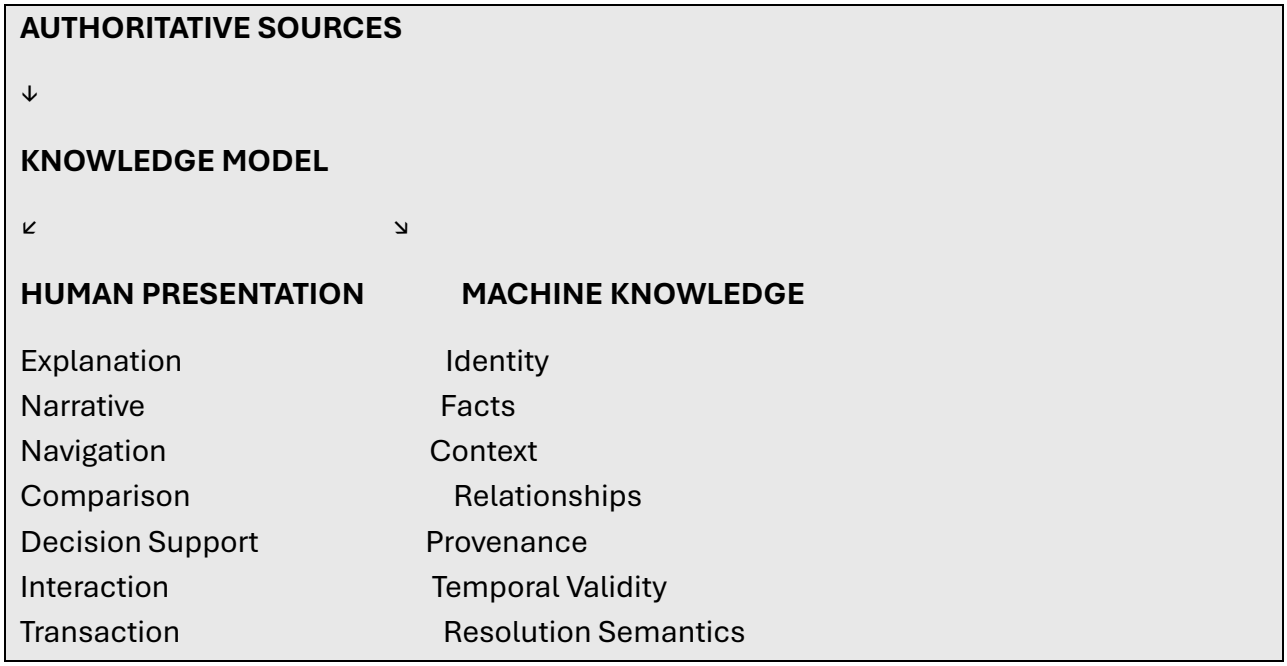
One is optimized for human comprehension.

The other is designed to make the components of the assertion explicit for machine interpretation and synthesis.

The architecture therefore does not require maintaining two independent versions of truth.

The preferred model is one authoritative knowledge foundation from which both representations can be produced.

Figure 7 — One Source of Truth, Two Renderings



This model is materially different from adding machine-readable metadata to a completed page as an independent publishing step.

The page is not necessarily the source of truth.

The knowledge model is.

The page becomes one representation of that knowledge.

The machine layer becomes another.

That distinction has operational consequences.

When an authoritative fact changes, an organization should not ideally need to update a paragraph, then a table, then structured data, then an API response, and then rely on editorial processes to keep those representations synchronized.

The architectural preference is:

Authoritative Source → Knowledge Model → Human Rendering + Machine Rendering

Update the knowledge.

Regenerate the representations.

Alignment becomes architectural rather than dependent entirely on editorial synchronization.

This does not require every organization to adopt one specific technical implementation.

It establishes a design principle:

Model meaning before rendering representations of it.

6.5 An Unexpected Benefit

The two-layer model suggests another consequence that emerged from the work but was not part of the original objective.

Separating machine-facing semantics from human presentation may allow human content to become more human.

Publishers have spent decades adapting human documents to the requirements of search systems.

Important terms are repeatedly defined.

Questions become headings because people search for them.

Facts are repeated across resources to establish topical relevance.

FAQ sections proliferate.

Writers are encouraged to make passages independently retrievable.

Generative search can intensify this pressure.

If every sentence is expected to function simultaneously as human communication, search evidence, a retrievable fragment, a potential citation, an entity signal, and a machine-interpretable factual object, the page begins serving too many purposes at once.

A separate Machine Knowledge Layer creates another possibility.

Writers can write for people.

Designers can design for people.

Product teams can build interfaces for people.

Machines do not necessarily need every important relationship reconstructed from prose if the relationship is represented explicitly in the knowledge layer.

They do not need every occurrence of a concept to carry another slightly different definition if canonical meaning has been established elsewhere in the architecture.

A human paragraph does not need to carry the entire burden of identity, context, provenance, temporal validity, and applicability if those properties are represented explicitly alongside the human resource.

This does not mean human content becomes less factual.

Nor does it justify placing contradictory or materially different claims in a machine-facing representation.

The principle remains alignment.

Each representation can specialize in the requirements of its consumer while remaining grounded in the same authoritative knowledge.

The Human Presentation Layer can communicate.

The Machine Knowledge Layer can formalize.

The knowledge model can keep them aligned.

This is the larger proposition behind the Two-Layer Web.

It is not a web divided between humans and machines.

It is a publishing architecture that recognizes that humans and machines can consume the same underlying knowledge differently—and that publishers can deliberately serve both.

The web page does not disappear.

It acquires another layer.

Human presentation above.

Explicit knowledge beneath.

Both describe the same reality.

Both originate from the same authoritative foundation.

Together they create a publishing surface designed for an information environment in which machines increasingly participate as consumers and interpreters of public knowledge.

Section 7.
WebMEM® — An Embedded Knowledge-Layer Architecture

The Two-Layer Web described in Section 6 creates an implementation question.

If a public web resource can contain both a Human Presentation Layer and a Machine Knowledge Layer, how might that machine-facing knowledge actually be published?

Where should it reside?

How should its semantic boundaries be represented?

How can identity, facts, context, relationships, provenance, temporal validity, and resolution requirements remain associated with the public resource to which they relate?

The production work underlying this study explored those questions through an architecture called **WebMEM®**.

WebMEM® is examined here as one implementation developed during the research. It is not presented as a required standard, a universal solution, or the definition of the broader publisher-side framework described in this study.

Its relevance is empirical and architectural: it provides a working example of how an explicit machine-facing knowledge layer can be embedded within the same public web resources used for human communication.

The human resource continues to explain, compare, guide, and support action.

Alongside it, an embedded knowledge layer represents aspects of what the publisher knows in a form intended for machine consumption.

The public resource becomes a carrier for both.

7.1 Framework vs. Implementation

A distinction is necessary before examining the architecture.

Knowledge Engineering for Answer Engines describes the publisher-side framework examined in this study.

WebMEM® is one implementation architecture developed through that work.

They are not interchangeable.

Knowledge Engineering for Answer Engines addresses the broader question:

How should authoritative facts, concepts, identity, relationships, context, provenance, and resolution rules be organized so that external generative systems can use them to construct answers?

That question exists independently of WebMEM®.

Engineered knowledge might ultimately be exposed through embedded structures, APIs, knowledge graphs, dedicated machine endpoints, emerging standards, agent interfaces, or architectures that do not yet exist.

WebMEM® addresses a narrower implementation question:

Can machine-consumable knowledge be published within the existing public web resource itself?

That distinction is important to the research.

The larger publisher-side problem should not depend upon the adoption or effectiveness of one implementation.

If other architectures prove more effective at representing contextual, provenance-aware, resolvable knowledge, the underlying problem remains.

The study therefore treats WebMEM® as a production implementation through which several of the broader architectural propositions could be explored.

The framework describes the knowledge problem.

WebMEM® provided one way to test it on the public web.

7.2 Embedded Knowledge

Machine-facing knowledge does not inherently need to reside within a web page.

An organization could expose it through an API.

It could maintain a public knowledge graph.

It could publish downloadable datasets.

It could create separate machine-oriented endpoints.

It could operate services through which agents retrieve authoritative information.

Each approach has different architectural and operational characteristics.

The WebMEM® implementation began from a particular hypothesis:

The public web resource itself can carry an explicit representation of the knowledge underlying its human presentation.

Consider a human-readable Medicare resource describing the coverage environment within a particular county.

The visible resource may contain:

- explanatory prose
- headings
- tables
- plan information
- definitions
- comparisons
- navigation
- decision support
- calls to action

A machine-facing representation of the same underlying information may instead make explicit:

- geographic identity

- applicable plan year
- available coverage categories
- plan counts
- plan-type counts
- premium observations
- relationships among concepts
- source datasets
- source versions
- provenance
- temporal applicability

The two representations serve different consumers while remaining associated with the same public resource.

This study refers to the implementation pattern as **surface-embedded knowledge**.

Conceptually:

ONE PUBLIC RESOURCE

↓

HUMAN PRESENTATION LAYER

-

MACHINE KNOWLEDGE LAYER

The implementation allows the resource to function as more than a human-readable document from which a consuming system must reconstruct all relevant meaning.

It can also carry an explicit machine-facing representation of the underlying knowledge.

Whether embedded knowledge is preferable to APIs, external knowledge graphs, machine endpoints, or future standards is not established by this study.

The relevant observation is narrower:

Embedded machine-facing knowledge is technically practical within the existing public web architecture and can be deployed alongside conventional human resources without replacing them.

WebMEM® provided the implementation used to examine that proposition in production.

7.3 Knowledge Objects vs. Fragments

The WebMEM® implementation also exposed an important distinction between the semantic architecture developed in Section 4 and the mechanics of publishing that knowledge.

A **knowledge object** and a **fragment** are not the same thing.

The knowledge object is semantic.

It answers:

What knowledge belongs together?

A fragment is a publishing and transport construct.

It answers:

How should some or all of that knowledge be represented and carried through the web?

The distinction prevents implementation mechanics from determining the semantic model.

Consider a knowledge object representing:

2026 Medicare coverage options in Mohave County, Arizona.

Its boundary exists because the underlying facts collectively describe one meaningful contextual state.

That object might be expressed through one machine-facing fragment.

Implementation requirements might instead make several related fragments appropriate.

Some concepts within the object may reference canonical definitions represented elsewhere.

Its provenance may reference reusable source information.

Relationships may connect the object to other entities or concepts.

The object defines what the knowledge **is**.

The fragment defines how that knowledge **travels**.

This leads to an implementation principle that became important during development:

Architecture precedes syntax.

The process should not begin with questions such as:

What tags should be used?

What JSON structure should be created?

Which properties should the markup contain?

The semantic questions come first:

What is being represented?

What is its boundary?

What is its identity?

Which facts belong together?

What context governs them?

What relationships matter?

What establishes their authority?

What limits their applicability?

Only after those questions have been addressed does the publishing representation become useful.

The markup is not the architecture.

It is an expression of the architecture.

7.4 Fragment Classes Observed in the WebMEM® Implementation

As WebMEM® developed through production use, different kinds of machine-facing information required different publishing structures.

These became known within the implementation as **Fragment Classes**.

The classes described here document the architecture used during the research. They should not be interpreted as a claim that these are the only possible semantic functions, that other implementations should use the same classifications, or that the taxonomy is complete.

They are useful because they show how the abstract elements developed in earlier sections were translated into a working machine-facing publishing layer.

Entity

An Entity Fragment establishes **what thing is being represented**.

Depending on the domain, it may represent:

- a product
- a Medicare plan
- a carrier
- a camera
- a manufacturer
- a drug
- a university
- a property

Its role may include stable identity, canonical naming, type, aliases, attributes, and relationships.

Its primary function is identity.

Dataset

A Dataset Fragment represents a coherent collection of contextual facts.

The Medicare Options implementation provides an example.

Its boundary is not determined merely by one source file. It is determined by the meaningful state the collection describes:

the Medicare coverage environment available within a defined geography and plan year.

A Dataset Fragment can therefore contain facts originating or derived from multiple authoritative sources while preserving their context and provenance.

Terminology

A Terminology Fragment establishes canonical meaning.

Instead of independently redefining concepts such as PPO, MOOP, formulary, deductible, sensor format, or lens mount every time they appear, a machine-facing layer can maintain a stable conceptual representation.

Other knowledge can reference that representation.

The implementation principle is:

Define meaning once. Reference meaning where needed.

Relationship

Some knowledge exists primarily between entities or concepts.

A Relationship Fragment can represent assertions such as:

Plan Segment → variantOf → Parent Plan

Camera → usesMount → Lens Mount

Plan → offeredBy → Carrier

PPO → typeOf → Medicare Advantage

The relationship itself is the assertion.

Where required, it can also carry context, provenance, or temporal validity.

Provenance

A Provenance Fragment represents authority and lineage.

Depending on the implementation, it may establish:

- authoritative source
- dataset identity
- source version
- publication date
- effective period
- methodology
- derivation

Multiple assertions or knowledge objects can reference common provenance rather than requiring source lineage to be independently reconstructed each time.

Resolution Requirement

A Resolution Requirement Fragment addresses the applicability problem developed in Section 5.

Conceptually:

Parent entity identified

↓

Multiple variants exist

↓

Relevant attributes differ

↓

Additional discriminator required

↓

Do not collapse variant-specific values

The purpose is to represent that candidate knowledge has been identified but that available context does not yet justify selecting one candidate as applicable.

This translates resolution sufficiency into an explicit machine-facing structure.

It allows the publisher to represent not only facts, but information about the conditions governing when those facts can safely be applied.

Figure 8 — WebMEM® Embedded Knowledge Architecture

PUBLIC WEB RESOURCE

- **HUMAN PRESENTATION LAYER**
  - Explanation
  - Narrative
  - Comparison
  - Navigation
  - Decision Support
  - Interaction
- **MACHINE KNOWLEDGE LAYER**
  - Entities / Identity
  - Facts / Datasets
  - Concepts / Terminology
  - Relationships
  - Provenance
  - Temporal Applicability
  - Resolution Semantics

The implementation does not require every public resource to contain every Fragment Class.

The semantic requirements of the underlying knowledge determine which structures are relevant.

A simple entity may require identity, facts, relationships, and provenance.

A contextual dataset may require substantial geographic and temporal context.

A terminology resource may primarily establish canonical meaning.

A resolver may contain relatively little explanatory material while requiring substantial identity and applicability semantics.

In that sense, the implementation follows the knowledge rather than requiring the knowledge to conform to one universal fragment structure.

7.5 What the WebMEM® Implementation Does—and Does Not—Establish

Because WebMEM® is the implementation used in the production work documented by this study, it is important to distinguish the architecture itself from conclusions that the research does not establish.

It Does Not Establish That Publishers Should Prewrite Answers for Machines

The WebMEM® implementation was not designed as a feed of canned answers.

Doing so would reproduce Question-Space publishing inside another representation.

The implementation instead attempts to expose knowledge from which different answers may be synthesized.

It Does Not Replace Human Content

The architecture assumes that humans continue to require explanation, narrative, interpretation, expertise, decision support, and experience.

The machine-facing layer exists alongside the Human Presentation Layer.

It does not eliminate it.

It Is Not Equivalent to Schema.org

Schema.org and other structured vocabularies can play important roles in identifying entities, properties, relationships, and characteristics of web resources.

WebMEM® was developed in response to a different implementation requirement: representing bounded bodies of contextual knowledge, including provenance and resolution semantics, within the public resource.

This study does not attempt to establish WebMEM® as a replacement for Schema.org.

The distinction is architectural rather than competitive.

A more detailed examination of Schema.org, structured data, and the requirements that led to the WebMEM® implementation is provided in the appendices.

It Is Not an Internal RAG Architecture

Retrieval-augmented generation generally addresses a problem on the consuming-system side:

How can a generative system retrieve supporting information from an available corpus?

The WebMEM® work examined here addresses the publisher side:

What explicit knowledge can a publisher make available to external generative systems it does not operate?

The two problems can intersect, but they are not the same problem.

It Does Not Define Knowledge Engineering for Answer Engines

WebMEM® is one implementation produced through the publisher-side research described in this study.

The broader framework does not depend on its adoption.

Other implementations may use different syntaxes, transport mechanisms, semantic structures, or publication architectures.

Future standards may address some or all of the same requirements differently.

The study therefore does not propose that every publisher should implement WebMEM®.

Its research value is that it provides a concrete architecture through which the Two-Layer Web, Knowledge Objects, Fragment Classes, provenance, contextual knowledge, and resolution semantics could be implemented and observed in production.

That distinction should remain explicit.

The enduring research question is not:

Will publishers adopt WebMEM®?

It is:

Should machine-facing knowledge be deliberately engineered as part of public publishing architecture, and what characteristics make that knowledge useful to external answer systems?

WebMEM® gives this study one working implementation against which that question can be examined.

It also preserves the division of responsibility developed throughout the preceding sections:

The publisher represents what it knows.

The answer engine determines how that knowledge participates in an answer.

The publisher does not need to predict every future question.

It does not need to script every future response.

It can instead attempt to make several things explicit:

This is what we know.

This is what it means.

This is what it describes.

This is how it relates.

This is where it came from.

This is when it applies.

These are the conditions governing its use.

The question can arrive later.

Section 8.
The Medicare Laboratory — Production Evidence

The architecture described in the preceding sections did not begin as an attempt to establish a new theory of publishing.

It emerged from production problems.

The work began with a practical objective: transform large quantities of authoritative Medicare data into useful public information for human and machine consumption.

The source data already existed.

It was authoritative.

It was machine-readable.

It contained stable identifiers, product information, geography, benefits, enrollment, performance measures, and other factual information.

Yet ordinary human questions frequently could not be answered safely by exposing those records alone.

Records from different sources had to be related. Geographic and temporal context had to be preserved. Product identities had to be understood at the appropriate level. Useful statistics had to be derived from defined populations. Terminology had to be interpreted consistently. Provenance had to remain connected to assertions and derivations.

And in some cases, apparently precise identifiers did not contain enough information to determine which known fact applied.

The production work therefore moved progressively beneath the presentation layer.

Data required context.

Context required identity.

Identity required relationships.

Assertions required provenance.

Derived facts required lineage.

And some information needs required explicit rules governing resolution.

Medicare became an unusually productive environment for examining these problems because the domain contained all of them at production scale.

The cases documented in this section are publisher-side observations. They do not establish the internal representations, retrieval methods, ranking systems, prompts, or reasoning processes used by external answer engines. The evidence is limited to what was published, what was queried, what appeared in generated results, which published facts were reflected in those results, and which public resources were surfaced as sources.

Within those boundaries, three production observations became particularly informative.

8.1 Why Medicare Was an Effective Laboratory

Medicare combines several characteristics that make it useful for studying machine-facing knowledge.

First, the underlying data is substantial and authoritative.

The Centers for Medicare & Medicaid Services publishes large structured datasets describing Medicare plans, contracts, geographic availability, benefits, enrollment, performance, formularies, and related program information.

The initial problem was therefore not a lack of data.

It was transforming source-oriented data into representations corresponding to the information people actually seek.

Second, Medicare is highly contextual.

A fact can depend on:

- plan year
- state

- county
- service area
- product type
- plan
- segment
- eligibility
- source version

A value without those conditions can be accurate in isolation and incorrect in application.

Third, Medicare contains hierarchical identity.

Contract IDs, Plan IDs, and Segment IDs can represent different levels of specificity. An identifier can appear precise while still referring to a parent entity containing multiple child variants.

Fourth, useful consumer information frequently does not exist as a literal field in one source file.

Questions may require derived observations such as:

- average premium
- number of $0-premium plans
- PPO count
- HMO count
- lowest maximum out-of-pocket limit
- carrier presence
- enrollment leaders

Those values must be calculated from authoritative records using defined populations and methodologies.

Finally, Medicare is a domain in which applicability matters.

A plausible answer is insufficient.

A value from the wrong county, wrong plan year, wrong product, or wrong segment may be an authentic CMS value and still be the wrong answer.

This combination made Medicare a useful production environment for examining the distinctions developed in the preceding sections.

Three cases were particularly important.

8.2 Case Study A — Medicare Options

The Problem

Traditional Medicare information architecture frequently follows product categories.

Medicare Advantage is one directory.

Part D is another.

Medigap is another.

Special Needs Plans are another.

That organization is understandable from the perspective of the underlying programs, source datasets, and publishing systems.

It does not necessarily reflect the information state of the person beginning the decision process.

A beneficiary may not yet know which product category is relevant.

The initial information need can be more fundamental:

What Medicare coverage options are available to me here?

That question suggested a different semantic boundary.

Instead of beginning with one product category, the production implementation constructed a county-level representation of the local Medicare coverage environment.

Conceptually:

Medicare Options → County → Plan Year

The resulting contextual knowledge object could contain information involving:

- Original Medicare context
- Medicare Advantage availability
- Part D availability
- Medigap context
- Special Needs Plan availability
- total plan counts
- plan-type counts
- PPO and HMO availability
- premium observations
- $0-premium counts
- maximum out-of-pocket observations
- carrier presence
- applicable geography
- plan year
- authoritative source information

The important architectural decision was not the number of fields.

It was the boundary.

Medicare Advantage, Part D, Medigap, SNPs, and related observations were grouped because collectively they described one meaningful state:

the Medicare coverage option environment within a defined geographic and temporal context.

The source datasets did not define the boundary.

The meaning of the information did.

The Observation

Following publication, the same underlying county-level representation could participate in materially different information needs.

One question could ask broadly about Medicare options within the county.

Another could focus on Medicare Advantage.

Another on PPO availability.

Another on Part D.

Another on average premium.

Another on $0-premium plans.

Another on maximum out-of-pocket limits.

The questions differed.

The underlying contextual knowledge remained the same.

The implementation did not require a separately authored answer object for every possible projection of the county's Medicare environment.

The relevant facts already existed within one coherent representation.

The information need determined which portion of that knowledge was relevant.

Same object. Different questions.

This provided a production expression of the Question Space / Knowledge Space asymmetry developed in Section 2.

A finite contextual representation could support information needs that had not been independently authored as answers.

This observation does not establish how an external answer engine internally represented the resource or which specific component of the published architecture produced the observed behavior.

It establishes something narrower:

One published body of contextual knowledge demonstrated utility across materially different information needs.

That observation materially influenced the subsequent publishing architecture.

8.3 Case Study B — Bacon County

The second case provided a different type of observation.

The Medicare Options work allowed substantially the same underlying factual knowledge to be independently rendered through two public web domains.

This created an opportunity to observe the behavior of a factual representation under conditions different from a conventional competitive search query.

Rather than asking a broad Medicare question for which many established ranking, authority, content, and demand factors might influence the result, an obscure factual question associated with a specific local knowledge state could be used.

Bacon County provided that test.

The underlying factual knowledge was the same.

The public resources were independently rendered.

An obscure information need corresponding to that knowledge was queried.

Both implementations appeared as evidence in the generated result.

Conceptually:

SAME FACTUAL KNOWLEDGE

↓

INDEPENDENT WEB RESOURCE A

-

INDEPENDENT WEB RESOURCE B

↓

OBSCURE FACTUAL QUERY

↓

BOTH RESOURCES SURFACED AS EVIDENCE

The observation was useful because the query was not an obvious conventional content target.

There was little reason to expect two independent public resources to have been developed around an editorial strategy targeting the same obscure factual combination.

The principal commonality was the underlying factual representation.

The evidentiary boundary is important.

The observation does **not** establish that Google created an internal knowledge object corresponding to the architecture described in this study.

It does not establish that Google preferentially retrieved WebMEM®.

It does not establish a particular ranking or retrieval mechanism.

It does not establish that every answer engine would behave similarly.

And it does not provide visibility into Google's internal representations, prompts, confidence models, retrieval architecture, or reasoning processes.

The publisher-side evidence is limited to what could be directly observed:

What was published.

Substantially the same underlying factual knowledge was represented through independent public resources.

What was queried.

An obscure factual information need associated with that knowledge.

What appeared.

Both implementations surfaced as evidence.

Which facts were reflected.

Facts corresponding to the published knowledge.

Which resources were surfaced.

The independently rendered public resources carrying that knowledge.

The observation is useful without requiring a claim about the hidden mechanism responsible for it.

It indicated that the factual representation demonstrated utility beyond one conventional document/query pairing.

The Bacon County case therefore complemented the Medicare Options observation.

Medicare Options demonstrated **projection breadth**: one contextual object participating across different information needs.

Bacon County demonstrated **factual reuse**: substantially the same underlying knowledge independently rendered and surfaced for the same obscure information need.

The next production case exposed a different boundary.

Something did not resolve correctly.

8.4 Case Study C — Plan-ID

The Plan-ID work initially appeared to present a simpler problem.

A Medicare plan identifier appears deterministic.

Input the identifier.

Retrieve the plan.

Expose the facts.

Conceptually:

ID in → Plan facts out

The underlying Medicare identity structure made the problem more complicated.

A parent identifier containing Contract ID and Plan ID can correspond to multiple Segment IDs.

Conceptually:

H5216-043

↓

H5216-043-1

H5216-043-2

H5216-043-3

Those segments are related.

They are not necessarily interchangeable.

And segment-specific attributes can differ.

This produced an observable failure mode.

An answer engine could locate the relevant plan family.

It could encounter the child variants.

It could retrieve values associated with those children.

Yet when the information request supplied only the parent identifier, the system could still lack the additional context required to determine which child applied.

In observed cases, the resulting generated response could combine child values into a range.

Suppose the child knowledge contains:

Segment A → $0

Segment B → $18

Segment C → $27

A generated response of:

$0–$27

contains no invented value.

Each number may correspond to the underlying authoritative data.

But if geography determines that exactly one segment applies to the beneficiary, the range has not resolved the information need.

The answer engine has located the candidate knowledge.

The remaining problem is applicability.

This exposed something the successful Medicare Options observations did not:

Retrieval success does not imply resolution success.

The negative observation was therefore as informative as the positive ones.

The problem could not necessarily be addressed by publishing another article.

It could not necessarily be addressed by writing another FAQ.

It could not necessarily be addressed by making the values easier to retrieve.

The values had already been found.

What remained absent from the interaction was sufficient information to determine how those values should be applied.

The publisher knew that the parent contained segments.

The publisher knew that segment-specific attributes could vary.

The publisher knew that geography could determine the applicable segment.

That domain logic already existed within the publisher's systems and understanding.

The resulting research question became:

Can the publisher expose that resolution knowledge too?

This led to the resolution requirement developed in Section 5 and represented conceptually as:

resolution_required

The intended meaning is not:

We don't know the answer.

It is:

The candidate knowledge is known, and we know what additional information is required before the applicable answer can be selected.

The Plan-ID case moved the research problem from retrieval toward resolution.

That distinction became one of the more consequential findings of the production work.

8.5 Three Production Lessons

Figure 9 — Three Production Lessons

MEDICARE OPTIONS

Synthesis / Projection

One coherent contextual knowledge object demonstrated utility across materially different information needs.

Same object → Different questions

BACON COUNTY

Factual Reuse / Retrieval

Substantially the same underlying factual knowledge, independently rendered, surfaced as evidence for an obscure factual information need.

Same knowledge → Independent resources → Same information need

---

PLAN-ID

Resolution Boundary

The relevant entity family and valid child facts could be retrieved while the applicable child remained unresolved.

Correct retrieval → Insufficient context → Resolution required

Taken together, the three production observations form a progression.

Medicare Options demonstrated that the answer does not necessarily need to be independently authored in advance.

Where sufficiently explicit underlying knowledge exists, materially different questions can request different projections of that knowledge.

Bacon County demonstrated that the utility of the factual representation could persist across independent public renderings.

The useful element was not necessarily one carefully optimized sentence or one specific page/query relationship.

Plan-ID exposed the limit.

Making relevant facts available does not establish that an external machine can determine which fact applies.

The production work therefore progressively moved the architecture beneath several initial assumptions.

At first, the problem appeared to be data.

But the data already existed.

Then it appeared to be structure.

But structured data alone did not make enough domain meaning explicit.

Then it appeared to be retrieval.

But successful retrieval did not guarantee applicability.

The production observations exposed the progression:

Data wasn't enough.

Structure wasn't enough.

Retrieval wasn't enough.

The architecture that emerged required a richer representation incorporating:

Facts

Context

Identity

Relationships

Provenance

Resolution

These are the elements developed throughout the preceding sections.

They were not selected first as a theoretical model and then applied retrospectively to convenient examples.

The direction of the work was largely the opposite.

Production problems repeatedly exposed domain meaning that the existing publishing architecture did not represent explicitly.

A county needed to become more than a page.

It became a contextual knowledge object.

A factual representation needed to support information needs that had never been individually authored.

It became an answer substrate.

An identifier needed to express not only identity, but the limits of that identity.

It became a resolution problem.

Each problem caused another element of domain meaning to become explicit within the architecture.

That is why the Medicare work is relevant beyond Medicare.

Remove CMS, Plan IDs, counties, premiums, and maximum out-of-pocket limits, and the underlying problems remain.

Products have variants.

Facts have context.

Entities have relationships.

Values have provenance.

Knowledge changes over time.

Identifiers have different resolution depths.

Applicability can depend on information the initial question does not contain.

Those are not uniquely Medicare problems.

They are general knowledge-representation and applicability problems.

Medicare made them unusually visible at production scale.

Section 9.
From Publishing Technique to Enterprise Practice

Once explicit machine-facing knowledge becomes part of the publishing architecture, responsibility for the problem begins to cross traditional organizational boundaries.

Search teams can improve discoverability.

Content teams can improve explanation and presentation.

Developers can implement machine-facing structures.

Data teams can expose and maintain source systems.

Subject-matter experts can validate domain meaning.

Governance and compliance teams can establish authority and acceptable use.

But none of those functions, operating independently, necessarily owns the complete publisher-side problem examined in this study.

Someone must determine which sources are authoritative.

Someone must decide what constitutes an entity or meaningful semantic boundary.

Someone must establish which facts belong together.

Someone must identify the context governing applicability.

Someone must define relationships.

Someone must preserve provenance.

Someone must recognize when an identifier is insufficient to resolve an information need.

Someone must translate those decisions into a machine-consumable representation.

And the organization must be able to test whether external answer engines are using the resulting knowledge accurately.

At that point, the problem extends beyond a publishing technique.

It becomes an enterprise information practice.

9.1 Enterprise Knowledge Already Exists

Most organizations do not suffer from a lack of data, expertise, or institutional knowledge.

The more common problem is fragmentation.

Useful knowledge may already exist across:

- databases
- APIs
- Product Information Management systems
- Content Management Systems
- ERP systems
- regulatory feeds
- product catalogs
- pricing systems
- research repositories
- operational applications
- internal documentation
- subject-matter expertise

A manufacturer may know every specification, component, compatibility relationship, configuration, revision, and market for its products.

A healthcare organization may know products, benefits, provider relationships, formularies, eligibility rules, service areas, pricing, and regulatory constraints.

A university may know programs, prerequisites, courses, faculty, tuition, campuses, application deadlines, accreditation, and degree requirements.

A financial institution may know products, rates, terms, jurisdictions, eligibility conditions, fees, and effective periods.

The knowledge exists.

But it frequently exists according to the operational boundaries of the systems and people that maintain it.

Product identity lives in the PIM.

Current pricing lives somewhere else.

Definitions live in the CMS.

Geographic availability lives in an operational database.

Regulatory requirements arrive through an external feed.

Source lineage lives in ETL documentation.

Applicability rules are implemented in application code.

Critical exceptions may exist primarily in the knowledge of subject-matter experts.

The organization can therefore know all of these things collectively without possessing a unified external representation of what it knows.

That distinction becomes important when the consuming system operates outside the organization.

An external answer engine does not have direct access to institutional memory.

It does not attend product meetings.

It cannot ask the database architect why two identifiers differ.

It does not inherently know that the compliance team considers one dataset authoritative over another.

It cannot inspect an employee's understanding of which product variant applies within a particular market.

It encounters what the organization has made externally available.

The enterprise problem is therefore not simply:

How do we expose our data?

It is:

How do we transform fragmented organizational knowledge into an externally usable answer substrate?

That requires more than transport.

It requires modeling.

9.2 The Enterprise Question

Organizations have traditionally measured digital publishing through units of output, discovery, attention, and business performance.

How much content have we published?

How much traffic does it attract?

Where do we rank?

How many pages are indexed?

How many leads or transactions result?

Generative search has introduced additional measurements.

How often are we cited?

How frequently does the organization or its information appear in generated answers?

Across which answer systems are we visible?

Those questions are useful.

But they remain measurements of the output surface.

The production work examined in this study suggests another enterprise question:

How much of what our organization knows have we made explicitly usable by machines?

The answer may be substantially less than the volume of information maintained internally.

An organization can operate sophisticated internal data systems while presenting external machines primarily with prose.

It can maintain detailed product relationships internally while publishing disconnected product resources externally.

It can preserve rigorous provenance within its data pipeline while removing much of that lineage from the public assertions produced by the pipeline.

It can enforce applicability rules inside an application while publishing resulting values without exposing the conditions governing their use.

From inside the enterprise, the knowledge can appear complete.

From outside, important elements of its meaning may remain implicit.

That gap becomes increasingly relevant as generative systems mediate how people encounter and interpret organizational information.

The strategic question may therefore become less about who can publish the greatest quantity of content and more about:

Who has made what they know most explicit, coherent, and usable?

9.3 An Emerging Knowledge-Engineering Function

The requirements identified through this study suggest an organizational function that crosses several disciplines traditionally managed separately.

The established term **knowledge engineer** is useful in describing some of this work, but this study does not propose a new professional title, redefine the established knowledge-engineering profession, or suggest that one individual should own every responsibility involved.

The more important observation is functional.

Publisher-side machine knowledge requires coordination across:

Domain Expertise

What is actually true within the domain?

Domain experts understand the facts, exceptions, applicability conditions, and distinctions that may not be evident from source records alone.

Data Architecture

Where does authoritative information reside, and how is it structured?

Data architecture establishes the systems, identifiers, transformations, and dependencies from which public knowledge may be derived.

Information Architecture

How should information be organized so that its meaning remains coherent?

The operational boundaries of source systems do not necessarily correspond to the semantic boundaries required for publication.

Information-Demand Analysis

What do people actually want to know?

Search queries, customer questions, support interactions, agent requests, and other forms of demand can reveal which parts of organizational knowledge need to be externally usable.

Semantic Modeling

Which entities, concepts, relationships, contexts, and constraints must be represented?

This determines the structure of the Knowledge Space.

Publishing

How should that knowledge be exposed through public resources?

The model must ultimately become externally accessible through some publishing architecture.

Machine Consumption

Can external systems retrieve and use the representation effectively?

Publication alone does not establish utility.

Governance

How are authority, provenance, consistency, temporal validity, and change controlled?

Machine-facing knowledge must remain aligned with the authoritative reality it represents.

The distinctive responsibility across these functions is translating between:

what the organization knows

and

what an external machine needs in order to use that knowledge correctly.

The Medicare Plan-ID problem illustrates the cross-functional nature of the work.

A content writer can explain the plan.

A developer can query the database.

A data engineer can expose Segment IDs.

A search specialist can identify information demand.

A Medicare subject-matter expert can explain why geography determines applicability.

But those observations must ultimately be reconciled into a semantic requirement:

The parent identifier is insufficient for certain attributes; child resolution is required before those attributes can safely be asserted.

That is a knowledge-modeling decision.

No single organizational title is required to make it.

But the decision must be made somewhere.

The function is not to personally know every fact.

It is to ensure that the organization's machine-facing representation preserves the meaning its systems and experts already understand.

9.4 Knowledge Modeling vs. Knowledge Compilation

The production work also exposed a practical distinction between two kinds of activity:

Knowledge modeling

and

Knowledge compilation.

They should not be treated as the same problem.

Knowledge Modeling

Knowledge modeling is primarily an intellectual and domain activity.

Humans determine:

- what constitutes an entity
- which facts matter
- which facts belong together
- what context governs them
- which concepts require canonical meaning
- which relationships exist
- which sources are authoritative
- what provenance must be preserved
- which temporal boundaries matter
- what resolution rules govern applicability

These decisions cannot generally be derived from a spreadsheet through syntax alone.

Software can detect that three segment records exist.

A human domain expert may need to establish that geography determines which segment applies.

Software can identify two different field names.

A human may need to determine whether they represent the same concept.

Software can calculate an average.

A human must define which records constitute the appropriate population if that average is to carry a precise meaning.

Knowledge modeling captures those decisions.

Knowledge Compilation

Once a semantic model has been established, many implementation tasks can become deterministic.

Software can:

- validate required fields
- normalize values
- enforce identifiers
- verify relationships
- attach canonical terminology
- bind provenance
- check temporal requirements
- identify incomplete objects
- generate machine-facing representations
- publish those representations consistently

This study refers to that process as **knowledge compilation**.

The distinction separates meaning from repetitive implementation.

Humans define the model.

Software validates, transforms, and renders instances of it.

This division is operationally important because machine-facing knowledge does not need to be manually authored object by object.

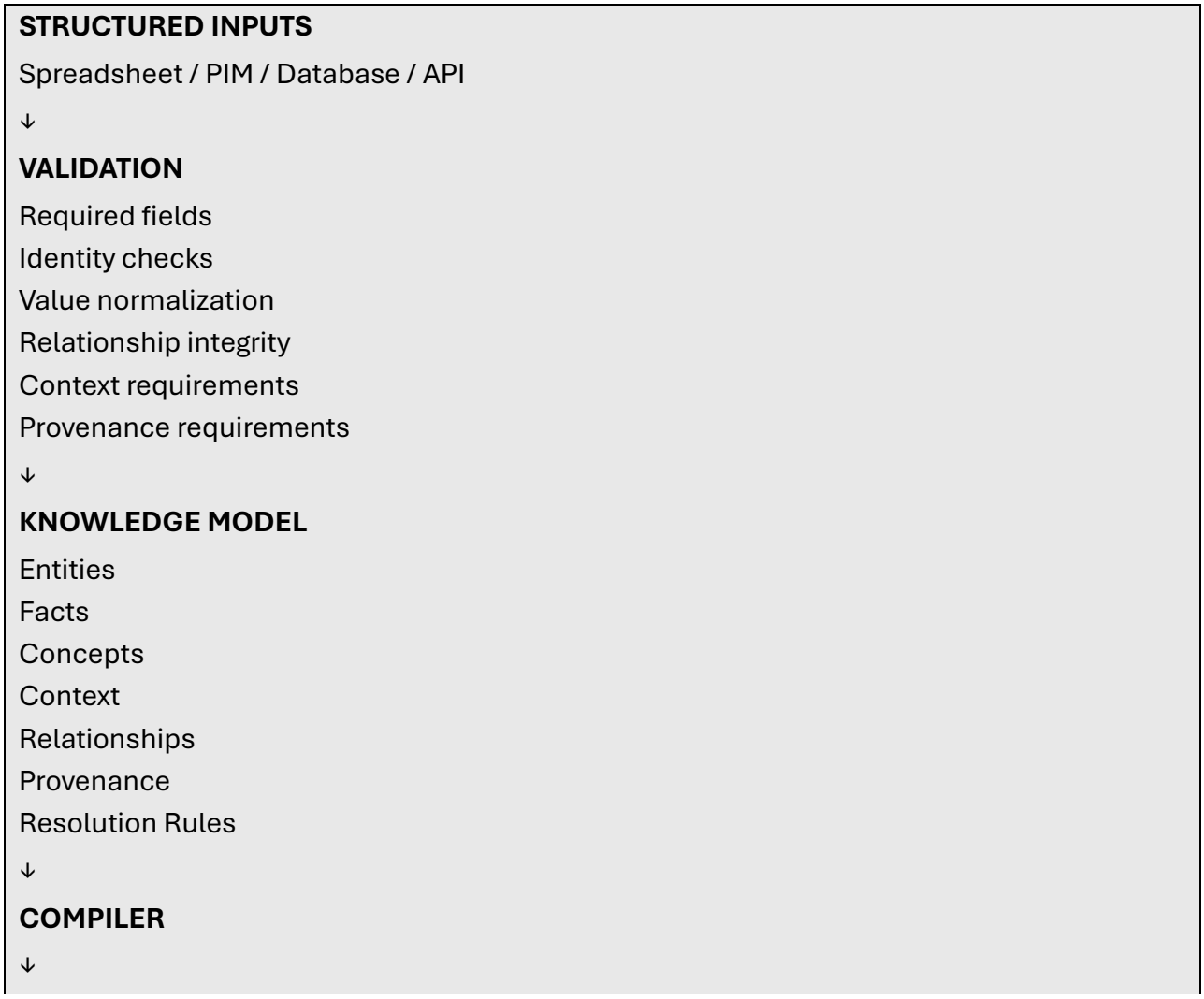
A domain expert should not need to write machine markup for 50,000 products.

A compliance specialist should not need to understand the transport syntax used to publish every representation.

A content strategist should not need to reproduce provenance metadata manually across thousands of resources.

Once the knowledge model is established, software can compile repeated instances from structured inputs.

Figure 10 — The Knowledge Compilation Pipeline



MACHINE KNOWLEDGE OBJECTS

↓

WEB PUBLICATION

This architecture separates meaning from mechanics.

The semantic decisions remain inspectable by humans.

The repetitive implementation work moves to software.

The particular machine-facing syntax can also change without necessarily requiring the underlying knowledge model to be reconstructed.

That follows the implementation principle established earlier:

Architecture precedes syntax.

9.5 The Spreadsheet as a Knowledge-Authoring Interface

The phrase *knowledge engineering* can suggest a large technical undertaking: an enterprise knowledge graph, specialized ontology platforms, semantic-web expertise, and a multi-year transformation program.

Such architectures may be appropriate in some environments.

They are not necessary to demonstrate the publisher-side model examined in this study.

A much simpler knowledge-authoring interface can be used:

a spreadsheet.

Consider a workbook containing several coordinated sheets.

Entities

What things exist?

entity_id	entity_type	canonical_name
camera_001	camera	Example Camera X

entity_id	entity_type	canonical_name
carrier_014	organization	Example Health Plan
plan_287	medicare_plan	Example Plan

Facts

What is asserted about them?

entity_id	attribute	value	unit
camera_001	weight	658	g
camera_001	sensor_format	full-frame	
camera_001	stabilization	true	

Terminology

What do recurring concepts mean?

concept_id	canonical_term	definition
ppo	Preferred Provider Organization	…
moop	Maximum Out-of-Pocket Limit	…

Relationships

How do things connect?

subject	relationship	object
segment_03	variantOf	plan_287
camera_001	usesMount	mount_04

Provenance

Where did the assertions originate?

source_id	source	version	effective_period
cms_2026_01	CMS	2026.01	2026
mfr_spec_44	Manufacturer specification	Rev. 4	current

Resolution

What conditions govern applicability?

entity	condition	required_context
plan_287	multiple_segments	geography
product_81	configuration_specific	model_variant

The spreadsheet is not the machine-facing knowledge layer.

It is a possible **human-facing authoring interface for the knowledge model**.

A compiler can transform these comparatively simple structures into whatever machine-facing representation the publishing architecture requires.

That separation is useful for several reasons.

Subject-matter experts can work in tools they understand.

Semantic rules remain inspectable.

Validation can detect errors before publication.

Implementation can evolve without requiring domain experts to relearn how knowledge is authored.

If the machine-facing representation changes, the underlying model does not necessarily need to change.

The compiler can change.

The spreadsheet example also demonstrates that the publisher-side framework does not inherently require a large knowledge-graph initiative before useful experimentation can begin.

An organization can start with a bounded domain.

Identify the authoritative sources.

Define the entities.

Define the important facts.

Establish context.

Model relationships.

Preserve provenance.

Identify resolution requirements.

Represent the model through ordinary structured inputs.

Validate it.

Compile it.

Publish it.

Test it.

Then expand if the results justify doing so.

The sophistication resides primarily in the **model and its rules**, not necessarily in the authoring interface.

This provides a practical path for organizations to investigate machine-facing knowledge without first restructuring everything they know.

The progression is:

INSTITUTIONAL KNOWLEDGE

↓

EXPLICIT KNOWLEDGE

↓

MACHINE-CONSUMABLE KNOWLEDGE

↓

PUBLISHED KNOWLEDGE

Once such a pipeline exists, the scaling question changes.

The organization is no longer limited to asking how many AI-oriented pages or fragments its publishing team can manually produce.

It can begin building a reusable, governed knowledge asset capable of supporting multiple public representations and information needs.

The same foundation may support pages, applications, agents, search experiences, APIs, or external answer engines.

The production work therefore suggests that the problem extends beyond search optimization or a single publishing technique.

It concerns how an organization operationalizes what it knows for external machine consumption.

Section 10.
Measuring Answer-Engine Utility

A machine-facing knowledge architecture is useful only if its performance can be evaluated.

This creates a measurement problem.

The metrics inherited from traditional search do not fully describe what happens when an external system retrieves published information and uses it to construct an answer.

A page can rank.

A source can be cited.

A brand can appear.

And the resulting answer can still be wrong.

An answer engine may retrieve the correct source but alter a value. It may preserve the value while losing the context that gives it meaning. It may retrieve several valid variants and select the wrong one. It may identify an ambiguity and then collapse that ambiguity into a definitive conclusion that the available information does not support.

Conversely, one knowledge object may demonstrate utility across materially different information needs even though no individual URL, keyword, or citation metric captures that breadth.

The production work examined in this study therefore suggests a different measurement question:

Did the knowledge work?

10.1 From Page Performance to Knowledge Performance

Traditional search metrics remain useful.

Publishers still need to understand:

- rankings
- impressions
- clicks
- traffic
- conversions

AEO and GEO introduce additional observations:

- citation frequency
- brand mentions
- generated-answer visibility
- source inclusion
- answer share

These metrics help establish whether a publisher appears within an answer environment.

They do not necessarily establish whether the publisher's knowledge survived the transition from source to synthesized answer accurately.

Suppose an answer engine cites the authoritative resource but reports the wrong premium.

Citation succeeded.

Knowledge performance failed.

Suppose it reproduces the premium correctly but applies a 2025 value to a question about 2026.

Factual extraction succeeded.

Context failed.

Suppose it retrieves three valid segment premiums and presents them as a range when the user's geography identifies one applicable segment.

Retrieval succeeded.

Resolution failed.

Evaluating machine-facing knowledge therefore requires measurement beneath the page and citation level.

The knowledge object provides a useful reference state for doing so.

The publisher knows what the object represents.

It knows the facts it contains.

It knows their context.

It knows the relationships among relevant entities and concepts.

It knows the provenance.

And where necessary, it knows the conditions governing applicability and resolution.

Generated answers can therefore be compared against the publisher's authoritative reference state.

This does not reveal how an external answer engine internally processed the information.

It provides a basis for evaluating the observable result.

10.2 Six Dimensions of Answer-Engine Utility

A complete measurement taxonomy could become extensive. For the production framework examined in this study, six dimensions capture much of the observable utility.

Presence

Does the publisher's knowledge appear in the answer environment for the information needs it should be capable of supporting?

Presence is broader than asking whether one URL ranks for one keyword.

If a knowledge object contains Medicare Advantage plan counts, PPO counts, HMO counts, average premium, maximum out-of-pocket observations, carrier presence, and other contextual facts, its potential utility extends beyond one narrowly defined query.

The measurement question becomes:

Across what portion of the intended Question Space does this knowledge object participate?

Presence indicates whether the published knowledge is being encountered.

It does not establish whether that knowledge is being used correctly.

Factual Fidelity

Does the synthesized answer accurately preserve the authoritative facts represented in the knowledge object?

An answer engine can cite the correct source while:

- changing a value
- misinterpreting an attribute
- combining incompatible facts
- deriving an incorrect conclusion

Factual fidelity compares an observable generated assertion with the publisher's authoritative reference state.

If the knowledge object establishes:

PPO count = 29

and the generated answer reports:

PPO count = 31

factual fidelity has failed.

The presence of a citation does not change that result.

The source can be correct while the synthesis is not.

Contextual Fidelity

Does the answer preserve the context that gives the fact its meaning?

A value can survive exactly and still become incorrect in application.

A generated answer might reproduce the source value while associating it with:

- the wrong entity
- the wrong geography
- the wrong year
- the wrong configuration
- the wrong jurisdiction
- the wrong population

Suppose $15.26 is the correct average Medicare Advantage premium for one county.

If an answer engine reproduces $15.26 but applies it to another county, the number survived.

Its meaning did not.

This creates an important distinction:

Factual fidelity asks whether the fact survived.

Contextual fidelity asks whether its meaning survived.

Projection Breadth

Across how many materially different information needs does the same knowledge object demonstrate utility?

Projection breadth is particularly relevant to the Knowledge Space model examined in Section 2.

A Medicare Options object may contain knowledge capable of supporting questions involving:

- total plan availability
- PPO availability
- HMO availability
- average premium
- $0-premium plans

- maximum out-of-pocket limits
- carrier presence
- Part D
- Special Needs Plans

These are not merely linguistic variations of one information need.

They are different projections across the same underlying knowledge.

Projection breadth should therefore measure **semantic breadth**, not raw prompt count.

Ten differently phrased questions asking:

How many PPO plans are available?

represent one underlying projection.

Questions involving PPO count, average premium, maximum out-of-pocket limits, carrier presence, and Part D availability represent materially different projections.

Projection breadth provides one observable measure of the principle:

Engineer once. Answer many.

A knowledge object demonstrating increasing projection breadth provides increasing information utility without requiring a corresponding increase in independently authored answers.

Resolution Accuracy

Did the generated answer select the applicable entity or variant?

Resolution accuracy is distinct from factual fidelity.

Suppose:

Segment A premium = $0

Segment B premium = $18

Both facts are authoritative.

The user's geography resolves to Segment B.

The generated answer reports $0.

The reported value exists within the source knowledge.

Retrieval may have successfully located the relevant family of facts.

But the answer is incorrect because the wrong applicable entity was selected.

That is a resolution failure.

A measurement framework limited to factual grounding can fail to detect it.

Resolution Restraint

Does the generated response avoid a definitive conclusion when the available context is insufficient to support one?

This is the least conventional dimension in the framework.

Answer systems are generally evaluated on their ability to provide answers.

The production work documented in this study exposed cases in which the more accurate behavior would be to **not answer definitively yet**.

If several plan segments exist and geography determines which one applies, a system that requests:

Which county do you live in?

may be behaving more accurately than one that confidently selects a premium from the candidate values.

Resolution restraint evaluates whether the observable response respects the boundary of valid assertion.

The desired behavior is not ignorance.

It is appropriate handling of uncertainty.

The relevant knowledge has been identified.

Several facts may be valid candidates.

But a missing discriminator prevents one candidate from being established as applicable.

Recognizing that condition is itself useful answer behavior.

Figure 11 — Answer-Engine Utility Scorecard

Dimension	Measurement Question
Presence	Is the knowledge encountered where expected?
Factual Fidelity	Are authoritative facts preserved accurately?
Contextual Fidelity	Is the context governing those facts preserved?
Projection Breadth	Across how many materially different information needs does the object demonstrate utility?
Resolution Accuracy	Is the applicable entity or variant selected correctly?
Resolution Restraint	Does the response avoid definitive conclusions when resolution is incomplete?

Together, these dimensions provide a more complete observational framework than citation frequency alone.

A knowledge object can be highly visible and poorly used.

It can be accurately quoted but contextually misapplied.

It can demonstrate broad utility while receiving relatively little conventional search traffic.

Or it can produce behavior that is especially valuable when ambiguity exists: recognition that additional information is required.

The framework does not claim to measure the internal reasoning quality of an answer engine.

It measures observable behavior against the publisher's known reference state.

10.3 Queries Become Test Cases

This measurement model also changes the role of query research.

Traditional search programs use queries primarily to identify publishing opportunities.

A query reveals demand.

Demand can justify content.

Within the publisher-side framework examined here, representative questions can serve another purpose:

They can function as test cases against the knowledge model.

For each representative information need, the publisher can ask:

1. What facts are required?
2. What concepts must be understood?
3. Which entities are involved?
4. What relationships must exist?
5. What context determines applicability?
6. What provenance is necessary?
7. Is resolution possible from the information supplied?
8. Does the knowledge model contain everything required?

A failed test can therefore reveal something more useful than the absence of a page.

It may expose:

- a missing fact
- a missing relationship
- missing context
- missing terminology
- insufficient identity
- missing provenance
- a missing resolution rule

The Question Space becomes a **test surface for the knowledge architecture**.

Instead of immediately asking:

Should we write something for this query?

the publisher can first ask:

Why can't our existing knowledge support it?

Sometimes the answer will be that the information need requires human explanation, analysis, or judgment.

Create that content.

But sometimes the question exposes a defect or omission in the knowledge model.

Correcting that defect may improve the architecture's ability to support not one query, but an entire class of future information needs.

This is one of the practical consequences of moving from Question-Space publishing toward Knowledge-Space engineering.

10.4 Measurement Closes the Engineering Loop

The production framework suggests an iterative operating model.

OBSERVE INFORMATION DEMAND
↓
IDENTIFY KNOWLEDGE REQUIREMENTS
↓
ENGINEER KNOWLEDGE OBJECTS
↓
PUBLISH
↓
TEST ACROSS ANSWER ENGINES
↓
MEASURE OBSERVABLE UTILITY
↓
IDENTIFY FAILURES
↓
IMPROVE THE KNOWLEDGE MODEL
↓
REPUBLISH

The important feature of this loop is where a failure leads.

In traditional content optimization, a poor result often sends the publisher back to the page.

Rewrite the heading.

Add a paragraph.

Expand the FAQ.

Improve internal links.

Those may remain appropriate responses to presentation, retrieval, or discovery problems.

A machine-knowledge failure can point somewhere deeper.

Perhaps identity is ambiguous.

Perhaps provenance is incomplete.

Perhaps a relationship is missing.

Perhaps temporal context is not explicit.

Perhaps the knowledge-object boundary is incorrect.

Perhaps resolution depends on a discriminator that was never represented.

The correction can then occur at the level where the observed problem exists.

Because multiple information needs may depend on the same underlying knowledge, one architectural correction can potentially improve many future answers.

The process becomes:

Build.

Test.

Measure.

Diagnose.

Improve.

Republish.

Test again.

This also establishes an important distinction between publication and continuing validation.

Publication asks whether the intended knowledge was represented and exposed correctly at a point in time.

Operational validation asks whether that representation continues to support accurate factual use, contextual use, projection, and resolution as the underlying knowledge and external answer environment change.

Source data changes.

Temporal validity expires.

Relationships change.

New information needs expose missing semantics.

External answer systems change.

A machine-facing knowledge layer should therefore be treated as a maintained information asset rather than a one-time publishing artifact.

The objective is not simply to increase the number of pages or citations appearing in AI-generated results.

It is to improve the observable reliability, applicability, and reusable utility of the knowledge those systems encounter.

Measure whether the knowledge works—not merely whether the page appears.

Conclusion:
Toward the Knowledge-Native Web

The production work documented in this study points toward a broader change in the relationship between publishers, machines, and information.

The web has repeatedly evolved by changing what machines can do with published information.

The original web connected documents.

Search engines made those documents discoverable.

The semantic web sought to make entities, attributes, and relationships more explicit.

Generative answer engines introduce another capability.

They can retrieve information, combine evidence, interpret relationships, apply context, perform derivations, and synthesize responses to information needs for which no publisher necessarily created a corresponding document.

The internal mechanisms by which individual answer engines perform these operations are outside the scope of this study.

The publisher-side consequence is observable.

For most of the web's history, publishers created information products for humans and machines helped people find them.

Generative systems can increasingly participate in constructing the information product itself.

That changes the publishing problem.

Publishers must consider not only the documents they create, but the knowledge external machines encounter when attempting to construct answers from those documents and the information beneath them.

The Answer Substrate

A central proposition emerging from this study is that a publisher does not necessarily need to publish every answer.

It can publish **what answers can be made from**.

If an organization possesses authoritative knowledge about its products, services, research, policies, markets, programs, or domain, one publishing strategy is to anticipate information demand and create individual resources intended to satisfy it.

Another is to make the underlying knowledge sufficiently explicit that multiple information needs can be supported from the same authoritative foundation.

Represent the entities.

Represent the facts.

Establish canonical meaning.

Bind facts to context.

Define relationships.

Preserve provenance.

Establish temporal validity.

Represent the conditions governing applicability and resolution.

Then allow external answer systems to perform the function that distinguishes generative synthesis from conventional document retrieval:

construct answers from available knowledge when questions arrive.

This suggests that the public web can function as more than a collection of documents waiting to be discovered.

It can also function as an **answer substrate**.

Not a database of prewritten answers.

Not a feed of canned responses.

A substrate of authoritative knowledge from which answers can potentially be constructed as information needs arise.

A More Computationally Useful Web

Documents remain extraordinarily useful.

They can be read.

Retrieved.

Referenced.

Quoted.

Interpreted.

Explicit knowledge can participate in additional operations.

It can be:

- filtered
- compared
- counted
- aggregated
- traversed
- resolved
- calculated
- combined
- synthesized

This distinction changes the relationship between publication and information demand.

A conventional publishing workflow often assumes that the information artifact exists before the information need can be satisfied:

Question anticipated → Answer authored → Document published

The Knowledge-Space model examined in this study makes another sequence possible:

Knowledge engineered → Question arrives → Applicable knowledge resolved → Answer synthesized

The publisher does not need to approximate the entire Question Space with documents.

It can concentrate on making the smaller, more stable Knowledge Space beneath those questions explicit and usable.

The production observations documented here do not establish that every information need can or should be served this way.

They demonstrate why the distinction deserves consideration.

Some information needs require authored communication.

Others can be projections across knowledge that already exists.

The Publisher-Side Responsibility

For decades, publishers became highly effective at creating documents.

Search created strong incentives to make those documents discoverable.

AEO and GEO have introduced new incentives to make information easier for generative systems to retrieve, cite, and incorporate into responses.

The work documented in this study exposes another publisher-side responsibility:

Make important domain meaning explicit rather than assuming an external machine will reconstruct it correctly.

That requires deliberate decisions about:

- authority
- semantic boundaries
- identity
- facts
- context

- relationships
- terminology
- provenance
- temporal validity
- applicability
- resolution

These are not solely content decisions.

They are knowledge-modeling and information-architecture decisions with consequences for public communication.

A relationship that exists only inside a database may not be explicit when an external answer engine encounters the related public resources.

An applicability rule implemented only in application code may be absent when the resulting factual value appears elsewhere on the public web.

Provenance preserved throughout an internal data pipeline may disappear when the resulting number becomes a sentence or table cell.

A distinction obvious to subject-matter experts may become ambiguous once information leaves the organization.

The enterprise can therefore know something without having made that knowledge explicitly available for external machine use.

As generative systems increasingly mediate information between organizations and people, that gap becomes more consequential.

The Strategic Question

The framework developed through this study suggests an enterprise question that extends beyond content volume and AI visibility.

Not only:

How much content have we published?

Not only:

How visible are we in generated answers?

But:

How much of what we know have we made explicitly usable by machines?

That question does not belong exclusively to search teams.

It does not belong exclusively to content teams.

It potentially involves technology leaders responsible for the systems in which organizational knowledge resides.

Data leaders responsible for authority, quality, lineage, and governance.

Enterprise architects responsible for how information moves across systems.

Product and digital leaders responsible for information experiences.

Domain experts responsible for meaning and applicability.

Publishers responsible for what the organization ultimately makes public.

And communications and marketing leaders concerned with how markets encounter and interpret the organization.

The boundary between internal information architecture and external communications architecture may therefore become increasingly important.

What an organization knows internally affects what it is capable of making explicitly available to external machines.

Write What Humans Need Written

Nothing in this study suggests that human-oriented content becomes less important.

A knowledge-native web should not become a web composed solely of machine records.

Humans still require explanation.

Narrative.

Analysis.

Judgment.

Instruction.

Persuasion.

Interpretation.

Expertise.

Decision support.

Those are communication problems.

They should be treated as such.

A specification can establish that a camera weighs 658 grams.

It cannot necessarily explain what carrying that camera for eight hours in the field feels like.

A dataset can represent factual differences between Original Medicare and Medicare Advantage.

It cannot necessarily help a first-time beneficiary understand the practical tradeoffs involved in choosing between them.

A product catalog can establish features.

It cannot necessarily provide expert judgment about which compromises matter to a particular buyer.

Human knowledge and human communication remain essential.

The distinction developed through this study is narrower:

Not every information need is a content problem.

Sometimes a new question requires explanation.

Write the explanation.

Sometimes it exposes knowledge the organization does not possess.

Acquire or create that knowledge.

But sometimes the information need is a projection across facts the organization already knows.

In those cases, another independently authored answer may add little.

The publishing challenge is to know the difference.

Let the Question Arrive Later

Perhaps the most consequential implication of the Knowledge-Space model is that the publisher does not need to know every future question in advance.

Instead, the publisher can establish:

what entities exist;

what facts describe them;

what concepts mean;

how things relate;

which contexts govern applicability;

where the knowledge originated;

when it is valid;

and what conditions must be satisfied before a definitive conclusion can be reached.

Represent those things coherently.

Maintain them.

Govern them.

Publish them in forms appropriate to the information consumers that need them.

Then allow the question to arrive later.

Tomorrow.

Next year.

From a customer.

From an employee.

From an analyst.

From an AI agent.

Or in a form nobody inside the organization anticipated.

If the necessary knowledge already exists in a sufficiently explicit and usable representation, the answer does not necessarily need to exist yet.

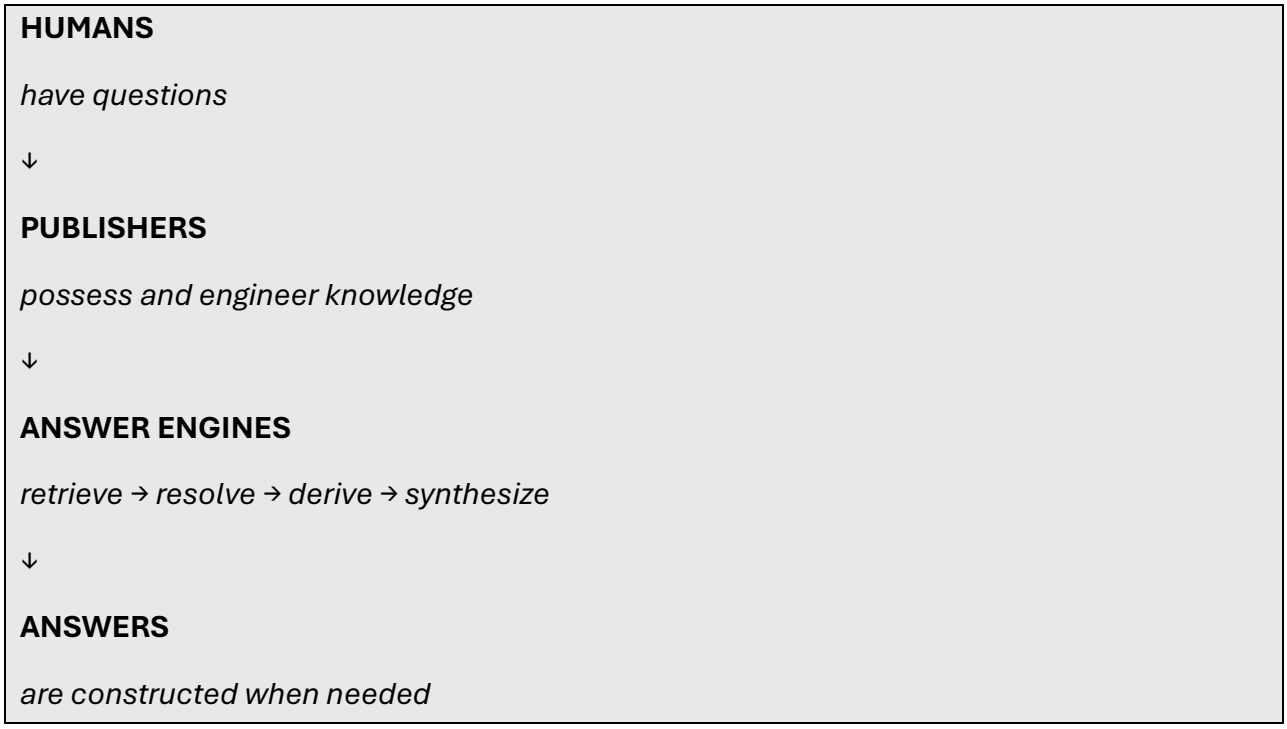
The question can establish the projection.

Context can establish applicability.

Resolution can identify the appropriate knowledge.

And an answer engine can attempt to synthesize the response.

Figure 12 — The Knowledge-Native Publishing Model



This is not the end of the document-centric web.

It is the emergence of another capability alongside it.

A web in which pages remain valuable human experiences while explicit knowledge associated with those resources becomes more computationally useful.

A web in which publishers do not necessarily need to manufacture every possible answer because they have represented the knowledge from which answers can be constructed.

A web in which the machine does not merely find what the publisher wrote.

It can increasingly work with what the publisher has made explicit about what it knows.

The architecture will evolve.

The formats will evolve.

The models will evolve.

Today's dominant answer engines will eventually give way to others.

The implementation developed in this study may evolve as well. WebMEM® represents one architecture explored through this production work, not a necessary endpoint for the larger idea.

The more durable relationship is simpler:

Humans have questions.

Publishers have knowledge.

Answer engines perform synthesis.

The publisher-side proposition examined by this study is therefore not that every possible question should be anticipated.

It is that the underlying knowledge should be represented well enough that every possible answer does not have to be prewritten.

Stop writing every answer.

Start engineering what the world needs to know.

Appendix A — A Working WebMEM® Knowledge Object
What does it look like?

Appendix B — WebMEM®, Schema.org, and Structured Data
Why isn't existing structured markup the whole answer?

Appendix C — Resolution in Practice
How does the difficult case actually work?

Appendix D — From Knowledge Model to Published Knowledge
How does an organization operationalize the architecture?

Appendix E — Building WebMEM® from Structured Data
Example code.