# From microcontent to neurons:

## A practical guide for building a cognitive AI content supply chain for highly personalized user assistance

by Michael J. Iantosca
Michael.Iantosca@avalara.com

2021-04-27

# Preface

This paper is a result of decades of effort by countless people. It's an audacious proposal to radically transform the content supply chain as we know it. It's not something that is built overnight; it takes a great deal of time and effort including the establishment of key foundational elements that includes content models, content types, content journeys, enterprise terminology, common taxonomies, and more. We know what all those elements are and how to create and manage them but make no mistake - it's no trivial undertaking.

Every year that passes, however, pushes the realization of a truly integrated, highly personalized, and proactive enterprise content experience that much further out – there are few shortcuts.

I've assembled what I consider a cookbook – a collection of recipes required to build a cognitive content supply chain. That vision was only a glimmer in the eyes of those that invented intelligent content. I was fortunate to have started my career at the dawn of that evolution nearly forty years ago at IBM where structured content was invented; I was mentored by the brilliant minds that invented it.

I do not profess to be an expert in some of the fields included in this roadmap. I know enough about cognitive computing to understand essential concepts due to the mandatory training I received at IBM with the development of IBM Watson™. I also have experience with Expert Systems (non-cognitive AI) that preceded it decades earlier, and I have respectable experience in the field of computational linguistics as well as several related patents and invention disclosures over the years.

What I do know is that designing and implementing a cognitive content supply chain is a team effort. It requires the buy-in and participation of a cadre of people that are experts in these fields.

My aim in writing this paper is to inspire peers and colleagues to join me in pursuing the lofty goals herein. The long and winding road may change course along the way, but I remain confident that we're on exactly the right path.

Regards,
Michael

"*Business, at the speed of trust*"

## Executive Summary

This paper describes the embodiment of a process and architecture that uses cognitive (AI) technology to drive an omnichannel content supply chain for integrated and highly personalized user assistance, at scale.

Content for user assistance is no longer limited for use as only customer enablement and support; content is essential for customer success, client time-to-value, and now more than ever, revenue. Enterprise user assistance content often resides in isolated functional silos. Such isolation inhibits a fully integrated and personalized customer experience; it remains a barrier to true *one-on-one* personalization. Moreover, the lion's share content often resides in the post-sales space. Post-sales content plays a direct role for prospective customers in purchase-making decisions.

Technical content generates more than 50% of viable sales leads (Forbes), is the second-most important pre-sales activity for technology buyers (IDC) and encompasses more than 55% of sales cycle time (vs. 21% spent talking to sales staff (Marketing Interactions).

> *Technical content generates more than 50% of viable sales leads, is the second-most important pre-sales activity for technology buyers and encompasses more than 55% of sales cycle time (vs. 21% spent talking to sales staff).*

For all these reasons and more, an *intelligent content supply chain* is needed. An intelligent content supply chain can bring the full spectrum of user assistance content together from across the enterprise. It can service any persona, at any point in the customer journey, in any channel, and in response to user signals to provide a dynamic, targeted, and highly personalized content experience.

Using a typical portal-based approach to consolidate content silos provides for only a tactical solution. We must completely reimagine content and the content supply chain to bridge enterprise content with the customer. We can do that with content modeling abstraction and patterns, also known as *content metamodels* and deep learning. This paper builds upon the efforts of countless people in the structured content discipline that's now commonly referred to as *intelligent*

*content*. Professionals in this space have been preparing for this new model of intelligent content management and delivery for years.

In brief, this paper proposes the following process and architecture to enable a cognitive content supply chain for user assistance. Please refer to the balance of this paper for a detailed explanation and discussion of each of the following elements.

1. Adopt an intelligent content strategy across all user assistance content domains by enabling all content assets as reusable objects. Break down content into the smallest reusable, but meaningful chunks for portability, reuse, repurposing, and dynamic machine-driven assembly. Wherever possible, encapsulate content objects in *self-describing* document object containers using structured languages such as DITA, Lightweight DITA, or other DOM representations such as JSON or equivalent methods. Content object Encapsulation can occur in the source or via nodal transformations. The goal is to achieve structured semantic containment of portable content objects. Those objects can be further enriched with additional semantic intelligence bound, directly or indirectly, to their object wrappers.
2. Define a consistent enterprise terminology corpus along with terminology sub-domains. Terminology sub-domains inherit from the enterprise terminology corpus to harmonize terminology across the enterprise.
3. Automate enterprise content consistency using an assistive computational linguistic service. Also, make all content across the enterprise consistent with grammar, writing style, and terminology using assistive computational linguistic services.
4. Perform an enterprise-wide mapping of content models at the element and object level. Such mappings are done by creating a content metamodel, which is essentially an abstraction layer that acts as a translator and orchestrator between concrete, domain-specific models. Use the resulting metamodel to identify gaps and align content models where possible. Alternately, use the mapping to perform nodal transformations. A content metamodel is also sometimes referred to as a *core model*.
5. Define a set of enterprise taxonomies for content object retrieval. At a minimum, create subject, purpose, content type, content journey, and content experience phase taxonomies. Define additional taxonomies useful

for personalized content retrieval and assembly that aid the generation of concrete ontological object relationships and inferred object relationships.

6. Harmonize taxonomies with the enterprise terminology corpus, again using assistive computational linguistic services.
7. Apply enterprise taxonomies for machine object retrieval to content object containers in the form of subject-predicate-object patterns (triples) to enable cognitive service retrieval and assembly irrespective of the storage location of content objects.
8. To automate the application and precision of taxonomy assignment at scale, train an autoclassification service. When the training of the autoclassifier sufficiently exceeds that of humans, use the autoclassifier to apply taxonomy labels to content objects as a batch operation. Use the interactive facilities of a computational linguistics service to assist and accelerate the training and precision of the autoclassifier. Continue to provide assistive services for the assignment of taxonomy values for newly authored content.
9. Further enrich the intelligence of structured content objects with additional intelligence in the form of content type, journey, and experience phase classifications. It is critical that the taxonomies defining these values be standardized and consistent across all content in the enterprise. Additional taxonomies that aid in the automatic generation of concrete and inferred relationships can also include audience, product, and other intents useful for cognitive services-based retrieval.
10. Adopt existing or generate a new ontology for the concepts that cover your subject domain(s). An ontology represents *concrete* relationships between content objects. Ideally, further changes to object metadata are reflected in the ontology and vice versa, establishing a self-maintaining model as each continues to evolve.
11. Generate a knowledge graph. A knowledge graph uses the ontology as its base and extends an ontology. Knowledge graphs are used by systems to automatically generate *inferred* and *predictive* relationships for intelligent content object retrieval.
12. Generate scenario graphs. A scenario graph represents known task patterns required to solve complex multi-task problems. A scenario graph aids machine processing to organize objects into collections of content objects,

and collections into scenarios without the need for a human to pre-define every possible prescriptive organization and navigation ahead of time.

13. Load the terminology, taxonomies, ontologies, knowledge graphs, and scenario graphs, and other assets into a cognitive service.
14. Query the knowledge graph using a knowledge graph query language such as SPARQL based on personalization data gathered from the user interface. The queries return individual objects as precision answers, and orchestrated collections of content objects for scenario-based solutions.
15. Optionally implement a hybrid blockchain model to amplify existing signals (utterances) or dynamically adjust to changing signals based on user behavior. The signals can then be used to refine, expand, or change the personalized content experience, accelerate cognitive system learning, or proactively assist the user with agents - with the user's permission of course.

The process of enabling dynamic, highly personalized user assistance for consumption and delivery by cognitive services requires a building block approach such as the one described herein. Doing so is a non-trivial undertaking, but no AI now or in the foreseeable future has the heuristics to organize, discover, assemble, retrieve, and deliver the right content at the right time, to the right person, and in the right experience without architecting an intelligent content supply chain augmented with intelligent content assets to do so.

## Prophets everywhere

If you've attended any of the countless webinars, conference sessions, papers, and other research on what has been coined *Intelligent Content*, you are likely intrigued with the broad array of possibilities. The most common of these are intelligent reuse and repurposing of content, single sourcing using a write-once, reuse-many model, improved search, and personalization.

Thousands of companies and organizations that have adopted an intelligent content strategy are enjoying all these benefits and more to one degree or another. Intelligent content is a rather simple notion. Intelligent content is modular, structured, reusable, separates format from presentation, and is semantically enriched such that the content is highly predictable for machine processing and automation.

> *Intelligent content is modular, structured, reusable, separates format from presentation, and is semantically enriched such that the content is highly predictable for machine processing and automation.*

It was not always called intelligent content. Intelligent content dates to the origin of structured content that began as Generalized Markup Language (GML) invented at IBM in the 1960s. GML became standardized as SGML in the mid-1980s and later morphed into XML. XML was made truly extensible with the Darwin Information Typing Architecture (DITA). Similar dialects such as JSON that also provide a Document Object Model (DOM) architecture came into existence later. JSON is typically used as an interchange format but does not include the inheritance model standard with DITA. DOM object models provide hierarchical content object containers that can be enriched with metadata. Such object models can be processed predictably by machines without a dependency on a traditional relational or object database.

This paper, however, is not about teaching intelligent content. Intelligent content has many purposes, but its most exciting applications have not yet been fully realized.

There has been a great deal of talk about personalization, bots, and uniting content across the enterprise - what I like to call content *silo-busting*. Several cCMS providers have made inroads on these, but they typically use conventional approaches that do not leverage intelligent content to their full potential. Why is that?

No one that has adopted an intelligent content strategy disputes the impressive benefits and value gained. However, most will tell you that they intrinsically know there are far more benefits to be had – many more, but there remains a lack of clarity on what those benefits are exactly and even less clarity on how to achieve the promise of cognitive-enabled content.

As a result, organizations forge ahead using conventional knowledge and existing commercial wares. I'm going to use a tired phrase here – a *paradigm shift*. It is a tired phrase because it was overused to characterize a seismic shift in a typical approach and a shift in the assumptions on how things are thought about or done. But I can think of no better phrase to describe what the balance of this paper proposes – a fully-functional cognitive content user experience, at scale.

## The broken supply chain

The main problem with the way most organizations approach content is to create content silos by function. Content silos evolve naturally and frankly, by necessity. The result is functionally optimized content that also creates a fragmented content experience. The fragmented content experience is the result of the content not being seamlessly retrieved from the various content silos. When you start thinking about solutions from the perspective of enabling seamless *intelligent retrieval* it all starts to make sense. But how many start solutioning from that point? Unless you architect content supply chains, few do.

Nevertheless, most shops come at solutions as one-offs. Need scalable single-source, reuse, and repurposing? Implement DITA. Need a bot? Build a one-off, then another, and another. Need to improve search? Try another search technology. Need to consolidate all your enterprise content into a single experience? Build a new website.

All these work *to some degree*, but most of them are one-offs, disconnected, and do not scale.

The problem is that organizations fail to approach content goals in a holistic, systematic way. We tend to use the tools and methods with which we are most familiar. You know the old saying - *when all you have is a hammer, everything looks like a nail*? Nothing better describes the way most organizations approach developing content supply chains.

Let us consider some common desired goals for content.

- Make content production fast and efficient
- Achieve content reuse and repurposing, at scale
- Achieve a write-once, reuse-many single-source strategy
- Improve organic search
- Improve portal search
- Personalize content
- Improve self-service with bots
- Enable cost-effective and speedy language translation
- Achieve omnichannel content delivery
- Improve content quality, consistency, and cognition
- Provide a seamless, integrated enterprise content experience

And just for the fun of it. Let us add a few truly audacious goals.

- Move from failure-mode content to *pro-active* user assistance
- Provide dynamic, automated, *one-on-one* hyper-personalized content
- Build scalable bots and agents with *precision answers* from a *single corpus*
- Deliver dynamic and personalized content for multi-task scenarios
- Enable automated cross-silo content discovery and reuse during creation
- Improve revenue through conversion rate optimization (CRO)
- Achieve autonomic user assistance (self-healing and adaptive[1] content)
- Robotic content generation, organization, and assembly

[1] Iantosca, Michael J. (Wake Forest, NC, US), Jenkins, Jana H. (Raleigh, NC, US), 2014, LINGUISTICAL ANALYTIC CONSOLIDATION FOR MOBILE CONTENT, United States, INTERNATIONAL BUSINESS MACHINES CORPORATION (Armonk, NY, US) 20140088953 I

The notion of shifting from failure mode content to predictive, proactive user assistance is a notion several of us conjured up as far back as the 1980s while we were developing some of the earliest electronic hypertext book technology around the time of the debut of the IBM personal computer. We knew what we wanted to achieve, but it would be decades until the technology existed to make it possible. At the same time, there were great debates over content chunking. Some brilliant folks were pioneering content object modeling using an object modeling system called Bachman long before content ontologies were a thing. Mind you, this was still the late 1980's and early 90's before the dawn of the web and when Expert Systems, an early application of non-cognitive AI, was emerging. All the notions were present – they all wanted to solve the Rubik's cube of providing a dynamic and personalized content experience at scale using content as *objects*.

This list provided earlier is only a subset of goals. The problem is how organizations approach achieving them. They most often attack each one-by-one, and often from the outside-in. Please do not misunderstand – the end-user experience always needs to be designed from the outside-in, but it is ultimately a dead-end from which to start when architecting holistic content supply chain and strategy to do achieve these things and more using a unified enterprise content model and enterprise content architecture.

Everything in that list can be boiled down to common patterns. The secret to making it work as a unified strategy and architecture is to build-up patterns from the inside out, patterns that cannot be constructed from the outside-in.

I have lost count of what I call *false prophets* in the content world. In the early days of word processing, it was all about using common templates and standard paragraph styles. In the early days of SGML and XML, many chased the elusive notion of the *Golden DTD*, then it turned to the *Golden Content Management System* which could act as a silo-buster - one that promised to rule them all! In practice, all it accomplished was to add yet another CMS to the ever-growing stack of content systems with few being retired. Finally, it turned to *Golden Web Portal* that would somehow magically provide a consolidated user experience from content portals that were multiplying like rabbits.

Almost every content consolidation model to date continues to make customers play the equivalent of *Where's Waldo* to find and assemble the content they need.

*Almost every content consolidation model to date continues to make customers play the equivalent of Where's Waldo to locate and assemble the content they need.*

What organizations need is a flexible supply chain that abstracts all the things we want to do with content – designed and built on metamodels where content becomes a service (CaaS). In a fully architected CaaS, content systems become a utility where content flows like electricity through a distributed grid. Content objects connect like neurotransmission in the brain that triggers synapses to pass messages between a network of cells. In our world, the equivalent of a neuron is the content object, and something called a *triple* that enables neurotransmission. We discuss these later.

*In a fully architected CaaS, content systems become a utility where content flows like electricity through a distributed grid. Content objects connect like neurotransmission in the brain that triggers synapses to pass messages. between a network of cells.*

Sounds blue sky, doesn't it? It's not. We have the technology and know-how to make content behave like neurons in the brain with the aid of cognitive systems. To realize it, however, we need to crush the notion that AI for content begins and ends with conversational bots. What is needed to solve the Rubik's cube of content is a *cognitive content supply chain*.

## Too big, and too small

When I began exploring notions about cognitive content, I asked what I thought was a reasonable question – had anyone tried to use a cognitive service on a large product documentation corpus? The response I got was revealing. I was told, "*We tried it, but the content was too big and too small.*" As I dug deeper, I learned that a team had turned cognitive services against our existing content corpus that

consisted of tens of millions of pages, and it wasn't effective in mining answers. It was then that I began to comprehend what *too big* and *too small* meant.

The documentation corpus was too big, not in terms of volume, but in the sense that the cognitive service could return only large blobs of content in response to queries. The resulting documents were too large. Given the size of the content collections, even when broken down into topics, would result in returning whole documents or whole topics. Even if it could return whole topics, a topic isn't a precise answer. Doing so effectively only narrows search results. As an industry, we continue to treat users as content archeologists and push failure-mode content models on them. Customers want only the information they need, no more, and no less.

*As an industry, we continue to treat users as content archeologists and push failure-mode content models on them. Customers want only the information they need, no more, and no less.*

So, what about that "too small" comment? The content had been authored for the most part as structured content and delivered as HTML in our web portal. The content was too small because retrieving only an element within a topic for an answer is insufficient in most instances. Content at every scope lacked sufficient semantic intelligence to perform automated machine retrieval and assembly into a coherent collection of content elements. Without the ability for machines to dynamically assemble content objects in a meaningful order to answer questions and deliver precise answers - let alone more complex multi-task scenarios - they were lost.

We need to *inject intent* into our content, upfront. Isn't that the very definition of intelligent content? Many of us have been doing exactly that, some of us for decades. So why haven't we reached the promise of cognitive content retrieval assembly by now? We haven't reached it because we haven't applied the *right* intelligence needed by modern cognitive services – most of us have only gone half the distance, but we've been on the right track all along.

## The cognitive content supply chain

What does a cognitive content supply chain look like and how do we build one?

We'll start with natively structured content to communicate the elements involved, but all content can ultimately be encapsulated as objects and enriched with the necessary metadata to participate. The primary difference is in the granularity of the objects and the degree and difficulty in making content objects semantically self-describing for automated retrieval, assembly, and delivery.

### Content objects

I much prefer to talk about componentized content as content objects. You see, the term *content component* has been somewhat abused. Those in the web portal space think about components as web page widgets, others consider a FrameMaker file a component of a book, those in the structured content world think about components as topics and topic collections, while micro-document evangelists consider sub-topic elements as components. We all have different notions of what constitutes a content component.

Instead, we'll discuss content as objects instead components. You'll understand why when we get to the ontologies and knowledge graphs. A content object is the smallest container of semantically enriched content that has a *useful purpose* and one that we can identify, retrieve, and assemble into some aggregate, channel-agnostic deliverable. A content object can be an XML element, a grouping of elements, a topic, a collection, an entire Word file, a support knowledge article, a PowerPoint presentation, or any binary large object (BLOB) such as video, a PDF, a graphic and so on. Each cannot be broken down any further as useful objects that we can identify, label, retrieve, and assemble into some aggregate form for consumption, independent of a particular delivery channel.

> *A content object is the smallest container of semantically enriched content that has a useful purpose and one that we can identify, retrieve, and assemble into some aggregate, channel-agnostic deliverable.*

Why is this important? To achieve our vision as content-as-a-service (CaaS) and the content supply chain as a utility to achieve all the goals we listed earlier, we need to generate *relationships* between content objects. Or put more succinctly,

we need to provide the right content intelligence for *machines* to generate those relationships.

To do it at scale, with "big data" as it is called, we need the ability to scale to hundreds of thousands or even millions of objects, and that requires the use of cognitive services to handle that scale.

### Non-granular content components

Not all content is created or managed initially as intelligent objects. A Word document, a PowerPoint, a PDF, a video clip, or any other content can *become* an intelligent object. All intelligent objects need to gain an object wrapper. The wrapper encapsulates the object and is enriched with metadata on the wrapper. Whether the object wrapper is XML, DITA, JSON, a relational database entity, or an object database entity, if the object is made *portable* and *retrievable* using its metadata, we can use it. A composite document is less granular than a natively structured or emitted XML, or JSON object, but it can become a reusable content object, nevertheless. Does it need to be an object in a DOM tree? No, but doing so can add valuable semantic relationships that an object would otherwise not have.

### Micro-documents

Encoding content objects in a content architecture such as a JSON, XML, or DITA provides a greater degree of granularity and semantic intelligence than other approaches. For example, documents encoded using DITA are constructed from collections of semantically self-describing elements. When enriched with the right metadata these become reusable objects called *micro-documents*.

We can even create subgroupings of elements within DITA topics that span containers outside of their normally constrained hierarchy using <DIV> elements to create reusable micro-documents for cognitive retrieval *without affecting their typical use in a DITA collection*.

Another intriguing notion with micro-documents is the prospect to perform both linear or non-linear assembly and delivery of micro-documents. As content professionals, we've been conditioned to specify the sequence of content in a prescriptive way. We were taught do to so when we first learned how and why to create outlines in grade school. With dynamic content object assembly, micro-documents open-up the potential for non-linear assembly of atomic content

objects by the machine in response to the *signals* received from users or machines. As an example, automated journalism (Robo-journalism) has existed for quite some time now.

### Reverse engineering the problem

I sometimes prefer to deconstruct complex problems in reverse from the intended goal. This is where many in our profession get stuck. We're often so focused on building solutions up like Legos from the bottom up and stop at a certain point or attempt to solve the puzzle from the delivery channel inward which are often bound to one specific delivery channel. The latter is the antithesis of how to achieve an omnichannel content architecture. We must engineer an omnisource content storage and *retrieval model* in parallel with an omnichannel content source and delivery model.

So where then *do* we start? We're going to start with object retrieval, and we're not going to use conventional search to do it.

### Cognitive processing

Let's examine the basic model of how conversational bots work. It is all about creating and managing a conversation between a person and the machine.

The machine initially receives a signal called an *utterance* typically through, but not limited to a conversational dialog interface. The utterance is processed to determine the user *intent* – often comprised of a verb and a noun. The processing is done by a cognitive service using *natural language processing* (NLP). The cognitive system breaks down the *signal* into *entities* (objects/nouns) and intents (actions/verbs). Then the machine retrieves and delivers one or more *responses* based on probability. If the signal is weak and the intent cannot be sufficiently determined, the bot may request more information to obtain a stronger signal from which to better determine the entity and/or intent. If the returned answer is not correct, or if the user selects among a choice of answers, the cognitive service creates an association between the utterance and the accepted or rejected responses. In effect, the system is constructing *relationships* just as humans learn based on the age-old learning sequence of association followed by assimilation.

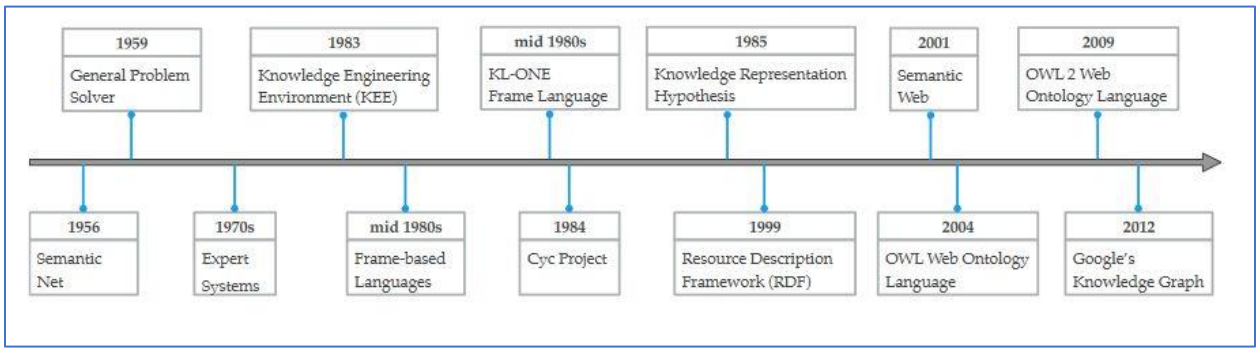
There are several ways to retrieve answers. A poor bot model uses a manually constructed decision tree. That might work for a narrow use case and a small

content corpus, but each is effectively a one-off. A more sophisticated model is for the cognitive service to use a map of objects and object relationships and return content objects that match user intent based on probability that improves with use.

Most of what we hear and read about with cognitive systems is how they can consume mind-boggling volumes of data and somehow instantiate those relationships automatically from unstructured data. They can, and it is amazing, but it requires an immense amount of human training. We can minimize the human training using enriched structured data instead.

## Ontologies

Greatly simplified, an ontology describes *concrete* relationships between objects. Ontologies are part of a linage of technologies called knowledge bases. An ontology formalizes the description of knowledge as an organization of concepts within a domain and the relationships that hold between them. Ontologies are often encoded in web ontology language (OWL) and managed by tools such as PoolParty™ or similar. The following graphic provides a brief history of knowledge bases[2].




For example, let's take three objects, Jimmy, Mary, and Harvard University. If we establish that Jimmy attends Harvard in one relationship and Mary attends Harvard in a separate relationship, the system can deduce that both Jimmy and Mary attend the same university, Harvard.

How does it do that? It's simple. Every node has three basic elements - a *subject*, a *predicate*, and an *object*. Sound familiar? Hopefully, it is, because that is how

[2] Ji, S., Pan, S., Cambria, E., Marttinen, P., and Yu, P. S., "A Survey on Knowledge Graphs: Representation, Acquisition and Applications", arXiv e-prints, 2020.

you were taught how to construct a basic sentence in English class. In ontology lingo, the combination of these basic three elements is called a *triple*. And it makes sense, doesn't it? If a natural language processor is at the heart of a cognitive system, then everything is based on the most fundamental mechanisms of language.

So what use are ontologies anyway? As we said, ontologies define concepts and concrete relationships between them. A cognitive system can use an ontology to validate and adjusts those relationships as it learns.

The ah-ha moment that finally hit me while deconstructing the problem occurred when I asked the most important question – *how does a cognitive service retrieve content*? It uses the mapped relationships, whether constructed by a human or inferred by the machine, to retrieve objects from where they reside and then deliver them to their intended destination.

It took me a while to figure that one out, not because it was difficult, but because like many in my field I had been overly focused on only the intelligent objects themselves for conventional reuse and repurposing. Making content objects intelligent by enriching them with metadata was a prerequisite, but the process of how cognitive retrieval works remained a sort of mystery. After that, all the pieces of the puzzle fell into place – and it is no small puzzle, but it becomes obvious after the general mechanics are understood.

## Triple threat

So, we now know that we can either have a cognitive system instantiate relationships from our content, or we can explicitly construct an ontology (or better yet, a *Knowledge Graph* (we'll get to those soon) and feed them to the cognitive system. The cognitive system then uses these structured *knowledge assets* as aids to determine which object(s) to retrieve. Simple right?

Earlier we noted how cool it is that a cognitive system can ingest a tome of unstructured content – say, every research paper ever written on a given subject domain like cancer, and automatically instantiate relationships. Make no mistake, the bigger the data, the bigger the network of relationships, which can be mind-numbingly huge, and why we need computers to construct and manage them.

But here is another little secret – not all data is unstructured. A cognitive system is more reliable and creates precise relationship when it uses *structured content*. It is far easier and more precise to construct cognitive systems from structured content than from unstructured content.

> *It is far easier and more precise to construct cognitive systems from structured content than from unstructured content.*

If your content is created and managed using a semantically structured document object model, congratulations - you've already objectified the content. As a result, all the content objects are *self-describing* containers with semantic element descriptors. Moreover, if you've enriched the content containers with useful semantic metadata in the form of attributes, then you are far ahead of the game. And if you've done it using an inheritance model such as DITA or Lightweight DITA, you're practically home free in terms of flexible granularity for retrieval.

What you have in effect is a structured object database, sans the database software. We don't necessarily need a database; the structured content *is* a database. Because DITA declares relationships through the DOM structure, inheritance, and edge knowledge in the form of semantically descriptive element wrappers and attributes, dare we consider DITA content a type of knowledge base? I do.

### Precision content

If you haven't yet made the connection, what we are constructing here has been coined *the semantic web* for quite some time, but I hesitate to use that term as the web is only one of many channels, and we're aiming for omnichannel content source management and omnichannel content delivery.

How do we make our objects even more precise? Well for one thing, when using encoding such as XML or JSON we can give our XML elements highly descriptive names because their elements are objects and can be made self-describing. Calling something a PartNumber for example rather than placing it in a generic list does wonders for machine processing. Collecting all the elements into a semantically typed topic also helps. Labeling a topic as a task, a concept, a reference, or even better, something highly specific such as an API, release note,

whatever, rather than just a generic topic, supplies *purpose*, which incidentally, maps back to – you guessed it, *intent*.

### Content containers

OK, so now we're far better off than a pile of unstructured documents. We have intelligent content, which is essentially structured as containers plus metadata. The notion of containerization cannot be understated. We are explicitly creating the containers using XML or DOM containers such a JSON or even XHTML. If our goal is to get as granular as possible when assembling personalized content by machine, then containers matter a great deal, the smaller the usable container, the better. Think of it this way - if you want the system to deliver only a subset of content to a user from a large document such as a list of steps to accomplish a specific task, the machine will need to figure out where the steps begin and end, and how much the steps depend on the content that precedes and follows. As good as cognitive systems are, they are far from having the heuristics to determine if the text preceding a list of task steps that described pre-requisites should be included or post-requisite information that follows the list of steps ought to be included. That is why DOM containers matter and why semi-structured content formats such as Markdown and ASCIIDoc fail miserably.

### Enriched content objects with more powerful semantic intelligence

We're not done with our content objects yet. Whether our containers are discrete structured elements, micro-documents, whole topics, whole topic collections, or whole Word documents, PowerPoints, or binary large objects (BLOBs), we can enrich them further with taxonomy labels.

Let's start with our granular structured content objects. We can define a set of key taxonomies that prove useful to construct essential relationships later as part of an ontology and knowledge graph. These models help the cognitive engine to do its job with a high degree of precision and less predictive guesswork before machine learning even begins.

A taxonomy is simply a hierarchical classification of things. If you've used eBay and navigated a subject tree, from photography to cameras, and then down to lenses you've seen and used a taxonomy. For our purposes, we need to define several taxonomies whose labels we can assign to our objects as attributes (metadata). We might define a subject and feature taxonomy for all the products

in our company's portfolio organized in a meaningful way. We might also define a purpose taxonomy to classify intended use.

Now that we have our taxonomies defined (no small task by the way for a large and complex content domain) we can assign the labels to our content objects as attributes or elements.

If that sounds like a daunting task, it can be, but we have ways to make it easier to do while creating content, and even more ways to automate the assignment of labels to objects in large volumes of legacy content.

### A "word" about taxonomies

Creating good taxonomies is no simple task, but a vital one. When creating taxonomies organizations should consider defining both enterprise and domain-specific taxonomies that are harmonized.

We need to keep our eye on the goal, however. That goal is enterprise-wide harmonization of content from across multiple domains to deliver a unified content experience to the customer and avoid perpetuating content silos. If we develop our taxonomies for only specific domains, it will be virtually impossible to create the integrated content experience we're after. All we'd be doing is isolating content silos further.

There is a place, however, for domain-specific taxonomies. That becomes self-evident if you have ever managed terminology at an enterprise-wide scope. What many organizations have learned is that there is a common core of terminology that spans the enterprise, but when you narrow down to the individual content domain level the terminology diverges to a degree – and understandably so.

The differences can co-exist, however. With advanced terminology management and computational linguistic systems, we already know how to create and manage hierarchies of terminology using terminology archetypes and domain sub-tree models.

After all, what are the primary elements of a taxonomy? Terminology! I am often stunned by how common it is that organizations isolate the design and management of terminology and taxonomies from one another. These must be *harmonized*, and we have the technology and methods to do exactly that with computational linguistic services, such as Acrolinx, Congree, and HyperSTE.

## And now, more words

Actually, a lot more words, about terminology.

The importance of managed terminology cannot be understated. It is a pre-requisite for content search to work effectively - let alone enable cognitive content retrieval. Yet formal terminology management is often ignored in many organizations at their own peril. Cognitive content processing is all about natural language processing. In a word - words.

One of the very first steps is to define a formal enterprise terminology corpus, ideally managed by one or more skilled terminologists. A typical termbase contains preferred terms, deprecated and prohibited terms, synonyms, and use-with-caution (context-sensitive) terms, definitions, usage rules, parts of speech, and more. It is not unusual for an enterprise termbase to contain thousands or tens of thousands of terms using a pivot language such as English, and hundreds of thousands more when national language variants are added.

Most organizations that have established termbases typically built them for localization providers, at least initially. Some make extended use of their formalized terminology by declaring a subset as simplified technical English (STE) to improve translation and reading levels. Although these are valid uses, it isn't enough.

Terminology consistency is of paramount importance to enable machine processing of content. Many organizations have discovered the critical role terminology plays for purposes such as effective search engine optimization (SEO) and comprehension - so much so that they've implemented computational linguistic services to manage terminology in the content creation and management phases. These assistive, rules-driven AI systems ensure accurate use of terminology in text, automatically flag incorrect or misuse of terms, and assist with corrections. The better ones ingest the organization's termbase; some can even mine terms and manage hierarchies of enterprise and domain-specific subsets of terminology.

Some organizations have embarked on enabling their content for cognitive processing only to stop everything and revert to harmonizing their terminology and fix other language inconsistencies across the enterprise corpus before continuing after discovering they've skipped this essential first step.

Computational linguistic platforms such as Acrolinx™, Congree™, and HyperSTE™ are among the common computational linguistic services in broad use. They might serve as the primary store for terminology or interface to a more powerful and feature-rich terminology management system (TMS).

There are several uses for applying managed terminology after it has been established and curated.

- Harmonize terminology across all participating content corpora both at the enterprise and sub-domain level.
- Harmonize enterprise taxonomies with the enterprise termbase.
- Use the termbase to assist with the training of a taxonomy autoclassifier.
- Use the termbase to help content creators assign the correct taxonomy values and metadata attributes to content objects.

It is confusing enough for humans when we use inconsistent and conflicting language. It is even more difficult for machines. Remember, one of the key constructs we care about most when architecting content for cognitive systems is the triple – subject + predicate + object. If we manage language then we improve the accuracy and consistency of our triples and the content itself, which in-turn improves precision, learning, and tuning of our cognitive content supply chain.

The same computational linguistics services also have rules-driven AI to normalize and make consistent the use of language in the content itself, including correct grammar syntax and writing style guidelines. These are highly advanced and configurable services that should be fully applied and considered a necessity, not a luxury.

### Content classification

Let us assume you've done the key pre-requisite to define your key enterprise taxonomies and maybe even domain-specific taxonomies along with your harmonized termbases - that's no small task.  Interestingly, this is the stage in which we find most organizations stalled. They intrinsically know that they are on the right track towards achieving cognitive content. Usually, they are stuck because they either haven't narrowed down the key set of taxonomies that truly matter that span the enterprise, or they are dealing with infighting and turf wars

between the content silos - unable to agree on common taxonomies and terminology. There is no easy way to combat this other than skilled or empowered leadership to mediate disagreements based on sound business decisions.

Nevertheless, let's assume you've conquered those barriers and we have our harmonized taxonomies. How do we do the enrichment of our content objects and not get stymied by the sheer scale of the effort? Remember, we are often dealing with mind-boggling volumes of content objects – even in small enterprises. In one shop, we had millions of product help pages. That was part of the "too big" side of the cognitive content equation. We can make it manageable, but how?

### Auto-classification and assisted classification

Enterprises have two core classes of content when it comes to applying labels - new content being actively authored, and legacy content that already exists, whether that content was created by the enterprise or was inherited as part of M&A activity – which for some companies, is a never-ending cycle.

Let's address the classification of legacy content first.

### Autoclassification

The very same cognitive technology that we want to use to perform dynamic and personalized content assembly has wonderful facilities that can assist us with making our content intelligent. It can do it efficiently and with far greater precision than can be done manually by humans. What is this magic of which I speak? Autoclassification.

Autoclassification uses state-of-the-art machine learning algorithms to suggest which labels from a controlled vocabulary are best suited to describe your content.

Cognitive services such as the IBM Watson Natural Language Classifier™ (NLC), IBM Watson Discovery Smart Document Understanding™ (SDU), PoolParty, and TopBraid™ as just some of the services that can perform autoclassification services with APIs[3]. I worked in one shop where an autoclassifier was used to do

[3] Manhaes, M., Ko, T., Selim, A., Amer, 0., Sri, L. Building Cognitive Applications with IBM Watson Services: Volume 4 Natural Language Classifier, 1985 IBM Redbooks

exactly what it implies: auto-assign taxonomy labels to content objects, at scale. Here is one way how it can be done.

Let's assume we have a taxonomy that has a thousand entries (labels). Organizations might assume they need to have humans manually assign taxonomy labels to content objects for all new and existing content and do it with a high degree of accuracy. Not so. Most shops wouldn't even add the necessary intelligence to legacy content; there are not enough human resources to justify doing so. But it is the existing content in which most of an organization's most valuable knowledge assets exist. So how do we deal with that barrier?

The first step is to train an autoclassifier. We can teach a cognitive engine how to apply labels, and to do it with far more precision than can a human, but first but we need humans to do the initial training.

1. Obtain access to an autoclassifier.
2. Feed a taxonomy to the autoclassifier.
3. Create or use the UI to analyze content and select suggested labels.
4. When the autoclassifier is sufficiently trained, turn it on the entire content corpus in to classify all the content using a batch process.

That doesn't sound too difficult now, does it? Although it takes a bit of effort to train an autoclassifier, the resulting efficiency and precision cannot be matched.

There are a few ways to approach training an autoclassifier. One way I've seen this work is to build a simple web portal into which content professionals manually cut-and-paste representative content samples. The portal can be connected to the autoclassifier using an API. The user pastes or imports content into the portal and the autoclassifier suggests labels from which to choose. The user selects the best label or rejects the suggested values and requests alternate choices until they select the most precise label. Which taxonomy is used doesn't matter; it could be a taxonomy that contains a thousand industry labels. Whereas a typical user could easily misclassify content, a trained autoclassifier rarely misses, often more than 90% or better, and that is also the degree of precision required for precision content retrieval and delivery.

## Assisted classification

What if we could aid the folks training an autoclassifier, and even better, assist them when classifying new content while authoring? We can.

There's a class of interactive computational linguistic tools available today that are gaining in popularity. These tools analyze content in either batch or in real-time when creating content.

While working with one of the leading computational linguistics systems a former colleague of mine and I had an idea – why couldn't we use that same technology to improve SEO? We wanted to mimic the algorithms used by popular search engines while actively writing content and help writers optimize the use of nouns and noun strings such that the search engines could more accurately rank the content in search engine result pages (SERPs).

The notion was simple. The computational linguistic tool was already capable of accurately gathering nouns and noun strings. All that was needed was to assign weights to nouns and noun strings using a flexible weighting algorithm. So, we collaborated with one of the computational linguistic providers to create a feature that:

1. Collected the nouns and noun strings.
2. Ranked them by *prominence* by scoring each against a weighting algorithm that factored frequency, density, and context.
3. Added an interactive UI to sort the results and optionally include ad-hoc strings not yet present in the content.

One embodiment of such a method is found in one of the computational linguistics systems called Acrolinx. The feature, called Findability, manifests itself as an interactive sidebar integrated with a wide variety of content authoring tools.

What if instead of using this capability to optimize SEO, we use it to optimize which taxonomy labels to select while training-up an autoclassifier to classify an existing content corpus or when manually applying taxonomy labels to new content? Let's examine these two potential applications closer.

## Using computational linguistics to accelerate training of an autoclassifier

Let's assume we want to train an autoclassifier using a representative sample of content from a given content domain. We've set up the autoclassifier, fed it a taxonomy, built a UI in which to input sample content, and we're getting recommendations from the autoclassifier from which to select values. Can we make that process more efficient to achieve the desired degree of precision more quickly? Yes, we can.

If we combine the noun and noun-string weighing services with the UI into which we're ingesting content to train the autoclassifier, the same algorithms used to analyze content for SEO can compare the taxonomy labels suggested by the autoclassifier and score them against the prominence ranking in the text. In a way, it's doing a similar analysis that the autoclassifier does, but given the weighting done by the computational linguistics tool that can factor in other weightings and terminology makes for an ideal assistive pairing. The autoclassifier, on the other hand, is designed for unstructured content, but in many cases, we have structured content, and our weighting algorithms are far more precise based on the structure mappings.

So why not then just use the computational linguistics service to do the autoclassification rather than the autoclassifier? For one thing, current computational linguistics tools are not cognitive; they don't learn and improve – at least not yet. That's why the pairing is ideal - each does something the other can't do or do as well.

Eventually, one of these commercial computational linguistics tools will integrate with a commercial autoclassifier and give us an off-the-shelf solution to use with a wide variety of content authoring tools via an integrated in-editor sidebar with no need to build an interface to an autoclassifier.

## Taxonomy assistance during the creation of new content

There isn't any reason why a similar solution cannot be used after batch autoclassification of an existing content corpus. The combination of suggestions from an autoclassifier combined with an in-editor sidebar that ranks and compares can be used to assign labels going forward while authoring. In effect, we can tool-up entire content teams to crowdsource the training of new

taxonomies for an autoclassifier as they create content right inside their authoring tool of choice while they write.

The other benefit of using computational linguistics service is to cross-validate the text of the object with the taxonomy labels. If the most prominent terms identified by the computational linguistics feedback indicate a significant mismatch with the suggested taxonomy labels assigned to the object, the mismatch becomes readily apparent. Either the suggested taxonomy value isn't correct or possibly the content itself needs to be tuned to better align with the assigned taxonomy labels.

### Exit the sidebar

We've diverted slightly from the main topic but solving the conundrum of how to classify both legacy and new content efficiently is essential to deal with the resource barriers needed to achieve it at scale with multiple taxonomies and a large corpus of legacy content.

### Beyond Taxonomies - Purpose

A fascinating question to ask is can we do something similar to perform autoclassification or assisted classification beyond nouns and noun-strings? Could we do the very same for auto-assignment of predicates to classify the *purpose* of the content?

A few years ago, I worked with an IBM team on a patent we called *The Gerund Momentum Principle*[4]. Simply put, it was a system based on an algorithm that weighted verbs against nouns and noun strings for a given topic. It used the results to automatically determine the content type and assign the correct content-type identifier. The identifier declares *purpose*. In a moment we'll discuss the role of triples needed to enable cognitive content. So far, we have our objects - the content containers which ideally are also semantically descriptive, and we've assigned the subject to our objects using the autoclassifier, but we have yet to complete the trio and assign a predicate to the object wrapper. This trio of metadata, the object, its purpose, and subject, creates the fuel that drives the

[4] George, Palliyathu Vishal (Babusapalya, IN), Iantosca, Michael J. (Wake Forest, NC, US), Kurian, John (Bangalore, IN), Sankar, Balaji (Bangalore, IN), 2019 UNSTRUCTURED DOCUMENT MIGRATION, United States INTERNATIONAL BUSINESS MACHINES CORPORATION (ARMONK, NY, US) 20190205460

rest of the cognitive content retrieval and delivery process that has proven elusive for so many and for so long.

### Honey, we're having triples!

Now before you faint, we're birthing triples, not triplets. Let's review what we have so far. We have containerized our content as objects and applied metadata to describe the subject and predicate of our objects. We have created the source for a triple right in our structured content source!

Not only that, but we can also do it at a micro-document level with content that is represented using a DOM structure. If all of this is seeming obtuse and is making you wonder why this is so exciting, stay tuned; we'll soon reveal the finale. Just keep in mind all those original goals we want to achieve with a single enterprise content architecture.

### Self-maintaining objects and ontologies and knowledge graphs

One intriguing prospect goes beyond the use of ontologies and knowledge graphs with our intelligent content objects. Not only are creating intelligent content but now we're sending our content to Harvard. Ideally, changes made to metadata on the content object source could be maintained bidirectionally with ontologies and knowledge graphs, keeping them in sync and self-maintaining. We can dream a little, can't we?

### Super objects

We know one of the goals we want is personalization. We can add additional metadata to our objects that can percolate up the ladder as attributes in our relationships.

We can and should classify our documents by content type using a consistent *content type taxonomy* across the enterprise. For example, a product guide is often called as many as five or more *different* names across an enterprise. Standardizing on one consistent naming convention for any given content type is essential for aggregating content from multiple content silos. It is not at all unusual to have dozens of overlapping names for the same collection of content types across an enterprise. By standardizing those names, we can add a reliable label such a "Position Paper" to the document object. When the engine looks for position papers it will find them – all of them.

## Content modeling using content metamodels

Content modeling for content types is common within and across domains. Models define the semantic sub-elements of each document type. They describe what each element *is*, not how it looks, along with the sequence and order of those elements plus any additional attributes.

Content models can be defined simply as guidelines for content creators, or they can be formalized through user interfaces such a forms and paragraph styles. Structured models most often encode the models in parser-enforced schemas. Formalizing content models is useful for standardizing the content for consistency and provides for predictable machine processing and automation.

Unfortunately, content modeling, if it is done consciously at all, is typically splintered across an enterprise, even for identical content types. That is understandable as the models typically originate within specific content domains. Not only does this create significant inconsistencies for customers who consume content from across the enterprise, but it also perpetuates the content silos themselves. Simply gaining enterprise-wide agreement on the names of content types is also an important first step but gaining consistency across elements within content models would be the ultimate achievement.

Let's illustrate the challenge for a common content type – a task. One department might have labeled their content element a "task", another department labels theirs a "procedure", still, several others use a different label. Why does this lack of consistency matter? It can matter a great deal when we attempt to combine content from across different domains for reuse, repurposing, and personalization, especially when using machine processing. Effective machine processing demands consistency and predictability.

Achieving fully aligned content models would be an exercise in futility for most organizations. As the saying goes, the cat is most always out-of-the-bag due to the evolution of content siloed by function. So how can we overcome that barrier and gain alignment? Metamodels.

A metamodel is an abstraction. We can map disjoint elements between similar and dissimilar content types or map them to a common archetype. The mapping can serve as a translation layer – element X in one document type is the

equivalent of element Y in a different document type. One company that specializes in this kind of content architecture is Simply[A]. Simply[A] has coined a name for such metamodels and calls them *Core Models*.

Certain document architectures, most notably DITA, avoid this problem using an inheritance model when deriving new document types. All DITA document types inherit their element definitions from a parent archetype and therefore share a common lineage. Machines can easily find commonality between child and parent content types via a DITA processing feature called *fallback* which the DITA standard provides through its inheritance mechanism.

There is also a sibling DITA architecture called Lightweight DITA that can "DITA-fy" other common content encodings. Lightweight DITA provides mappings between diverse content dialects such as XML, HTML5, and Markdown. However, it is not realistic to expect the use of a single content encoding for all content across an enterprise (we can dream though, can't we?) In any event, the more we can do so, the better.

At a minimum, a core model can be used to analyze and expose the mappings, and where feasible, align them. It can expose gaps and inconsistencies within and between content models. Moreover, it can be used to normalize the incongruences for machine processing – providing a virtual dictionary of content types and element synonyms to disambiguate the disjoint models where the machine processing needs to establish concrete relationships such as in ontologies and inferred relationships in knowledge graphs. Reduction of such ambiguity can aid in the training, learning, and precision when content is processed using cognitive services.

Several questions remain, however. Is it better to simply cross-map content models or map them to a single, abstract archetype such as that which DITA provides? Should such models be used to transform disjoint content types (that is, perform nodal transformations) and store them in a common *Content Lake* to align and optimize them for machine processing, retrieval, and delivery?

Innovative new tools are emerging that can assist with the analysis and generation of core models. They'll provide an efficient way to govern content models across the enterprise. Ideally, they'll be able to ingest existing schema to ease the effort and assist with the identification where the models do not align.

We might even directly interface such tools with a computational linguistic service to suggest model re-alignment, where feasible.

### Content journeys and temporal intelligence

A key reason why we standardize the names of content types across the enterprise is so that we can subsequently map the content types against our user journeys. Certain document types are needed by users at various points in a journey. The content needed at each phase is an aggregate of content types from different content domains. For software, content domains might include collateral from marketing, sales, partner enablement, product help, learning materials, developer enablement, and product support for example. A mapping of which content types are needed in which phase of the customer journey is critically important to the success of the customer. For a truly personalized user experience, our system needs *temporal intelligence*, which fulfills the "…*at the right time*" portion of our mantra. But how many systems do you know of that give little more than lip service to that aspect?

Worse, a customer journey is often only an organizational planning model for content. In actual use, customers typically bounce around phases of the journey. The actual use model by customers is rarely a linear one. For example, it is more common for a prospective customer to seek information about blockchain and start their purchasing journey in the product use phase than initially land on our marketing pages. An actual customer journey in action looks more like spaghetti than our neat but fictitious content journey sequence.

> *An actual customer journey in action looks more like spaghetti than our neat but fictitious content journey sequence.*

Modeling the customer journey is also one of the essential first steps required to enable a truly personalized experience. However, we'll need the power of cognitive services and utterances to break away from the bounds of prescriptive journeys and failure-mode content. We intend to provide customers with proactive and dynamically responsive user assistance based on what they are doing at any moment in time.

Also, without a consistent dictionary of content types and a dictionary of the user's journey, how would one determine whether there are gaps or duplication

of content needed for each phase of the journey? Moreover, how would a machine determine that?

### Journey and experience phases

A typical content journey sequence for software might look like the following.


…or some variation thereof. Juxtaposed against the content journey is the content experience phase against which we would do the same modeling. The content experience phases often overlap the content journey phases. Common experience phases in the use of software typically include the following.


So, why is this relevant to our object model? If we know some information about the user and can deduce what part of the content journey or experience phase they are in, we can personalize which document objects or collections to deliver.

For example, explore and evaluate overlap the discover, learn, and try phases of the customer journey. Sadly, many organizations stop at the modeling phase. These phase labels provide valuable metadata we need to assign to our content objects for a cognitive service to use for the temporal aspects of content retrieval and delivery.

If we add the content journey and experience phase labels to our content objects, our resulting knowledge graphs are that much richer and are better able to retrieve the right content, to the right person*, at the right time*.

We must give the machine enough intelligence to at least narrow down which content is appropriate and when. Now you know why adding temporal attributes, such as a content journey phase and experience phase is equally as important as the core elements of a triple.

## Our story continues to evolve

If you've read between the lines, we're following the basic model of storytelling – the basic model taught in elementary school to solve problems up through college journalism classes - we're designing a *Who, What, Where, When Why*, and *How* model enabled for machine automation. In business, we sometimes add a seventh element, *how much?* But I digress. It's a bit comical actually; here we are architecting for some of the most advanced technology of our time and most of what we are trying to accomplish with intelligent content boils down to what has been taught in elementary school long before computers existed.

## Let's review

We have done the most difficult work up until now. Let's review what we have thus far. We've

1. Defined our audience.
2. Standardized our enterprise terminology.
3. Harmonized our content with our enterprise terminology.
4. Standardize the definition of content types across the enterprise.
5. Defined our temporal models.
6. Broken down our content into granular, reusable components.
7. Encapsulated our components in semantically descriptive objects.
8. Defined and standardized our enterprise taxonomies and harmonized our taxonomies with our terminology.
9. Semantically enriched our content with:
    a. *who* (audience)
    b. *what* (subject)
    c. *where and when* (journey and experience phase)
    d. *why* (predicate)

If only our content could talk, and that is exactly what we're doing; we are giving voice to our content.

*If only our content could talk, and that is exactly what we're doing; we are giving voice to our content.*

## False prophets

This is the stage where many brilliant content strategists and industry experts stop when talking about intelligent content. They rightly evangelize the importance of structured content and metadata, content modeling, content journey mapping, standardizing terminology, defining taxonomies, and more. However, they often punt to commercial delivery offerings. They expect the content management system will somehow take the intelligent content and magically provide a responsive dynamic, highly personalized cognitive content experience of which we all dream. To be fair, there exists a small handful of commercial delivery wares that do a respectable job by overlaying business rules and act as a sort of content delivery middleware, but precious few are designed to interface with our custom models based on using *intelligent content assets* that complement the content – let alone use cognitive services.

### The golden content management system

Organizations that seek to combine content from across multiple heterogeneous silos often try several strategies. One of those strategies is what I call the fictitious *golden content management system* based on the golden schema or standard templates – these approaches only guarantee failure with only partial content consolidation. More often, attempts to address the problem with the next greatest CMS – you know, "the one that will rule them all!" often only multiplies the number of content management systems in an organization. I once was involved in an enterprise-wide CMS consolidation effort. By the time we finished taking inventory, we had uncovered dozens of content repositories, many of which we were not aware even existed. Even so, none of us were certain that we had found them all. Worse, and they kept coming – it was virtual whack-a-mole.

### The golden search strategy

I cringe every time I come across an organization that wants to provide a highly personalized content experience for their customers, and even moreover when they want to do it at an enterprise scale that spans all content. Inevitably the discussion turns to (drumroll please), improved search. We are left to wonder - what specifically are they are searching for and what do they expect to find and deliver with typical search services, and how? These are perfectly valid questions that we should ask and challenge those that would proffer solutions that start at the glass. There are "personalized" content experiences that really are not at all

personalized and then there are *highly personalized* one-on-one content experiences sometimes referred to as *hyper-personalization.*

With astonishing predictability, conventional search is where many shops focus their effort - at the glass. Conventional search strategies provide only tactical improvement and don't move the needle significantly in the overall scheme of things. You can add using conventional search technologies as a magic bullet to the heap of "the golden *whatever*" list of fictitious panaceas. What we really need is *search and retrieval* that learns and improves with use.

> What we really need is *search and retrieval* that learns and improves with use.

### The golden canonical delivery portal

Another tactical but dead-end strategy is attempting to consolidate content from across the organization using a canonical delivery portal strategy. It's not a bad tactical solution to placate senior leadership for a while, but you win no cigar for doing so. Canonical delivery portals are by their very definition dependent primarily on a single channel when we're aiming to enable a true omnichannel content delivery strategy. The best that typically results is the centralization of multiple, but separate portals mashed-up on the same real estate with enhanced search that attempts to act as the glue. But it lacks so much as you are beginning to realize. There is no path from there to true cognitive content – to a truly dynamic, highly personalized content experience. No matter how you dress it up, it remains an enhanced failure-mode assistance model that continues to force your users to solve the content puzzle.

Worse, if they've committed the next typical sin – adding intelligence *at* the *delivery channel* instead of enriching the source. they are unknowingly locking themselves into a specific channel; they've more than likely deluded themselves that a specific delivery channel can be truly canonical – yet another one to add to the magic-bullet trash heap.

## Cognitive content retrieval

I hope you are about to experience the "*ah-ha!"* moment as I did when I recognized a potential runway on which to land this jumbo liner. Again, many of

the folks in the intelligent content industry are amazing content experts, but only a few have hands-on experience designing and building end-to-end content supply chains - let alone a cognitive-based system. They defer to whatever commercial content solutions exist, especially on the retrieval and delivery end. This explains why many organizations remain mired in creating intelligent content but are not *using* it – at least not to its full potential. Content visionaries intrinsically know cognitive content delivery will become a reality someday but are waiting for Godot without doing the requisite foundational work.

*Content visionaries intrinsically know cognitive content delivery will become a reality someday but are waiting for Godot without doing the requisite foundational work.*

## Mapping the future

So, what do we do with all that content intelligence now that we have it? We map it, or better, have intelligent systems map it for us using our intelligent content objects.

As I've said, architecting cognitive content retrieval doesn't begin at the glass, it is all about automated content retrieval and assembly enablement *at the source* – for, and independent of, *any* channel.

So how we do cognitive content retrieval? *We* don't. The cognitive engine does. Which begs the next question - how do we enable it to do that?

As I said, you have already done the difficult part of turning your content into intelligent objects. What matters next is how to index into our intelligent content store – be it any store, and for our purposes multiple content silos from which to retrieve and assemble our objects into aggregate *knowledge*.

Earlier we discussed ontologies. Recall that we said that ontologies differed from taxonomies? Taxonomies bring formal order to the identification of objects; ontology is the identification of relationships between objects. It is not enough to simply use taxonomies to retrieve individual objects but rather retrieve a collection of *related* objects. To do so we need a mechanism that tells the machine what those relationships are.

Ontologies are commonly encoded using the Ontology Web Language (OWL). Just for clarification as these terms can get quite confusing, OWL is an ontology language while SKOS or better, SKOS-XL is used for creating controlled vocabularies, taxonomies, and thesauri. Some tools that can be used to generate these structured assets include those from providers such as SmartLogic™ and PoolParty™.

In theory, it would be ideal if we could generate an ontology core from our enriched structured content[5]. However, it is more common to develop an ontology using ontology development tools such as those already mentioned. An ontology is made up of triples, and the triples are stored in something called a triplestore (also called an RDF store). A triplestore is a database for the storage and retrieval of triples through semantic queries using a graph query language such as SPARQL (pronounced "sparkle").

### How ontology is used

We're finally getting closer to pay dirt. We now have our objects and our relationship model. But we still have not closed the loop of getting the content, let alone automatically organizing it.

You see, the size of the object store can become big very quickly – *really* big. There needs to be a highly efficient way to process the relationships and hash through the content stores to index into and content object stores at lightning speed. That is how a cognitive engine uses an ontology – to index into the content stores and get the content from a sea of objects. Cognitive systems continuously learn and self-tune relationships as it is used and responds to utterances and choices made by the user. We're not talking about only Bot input; we're talking about real-time content personalization.

A basic axiom of communication taught to every college Communications major on day one is fundamental – *you cannot not communicate*. That axiom was one of the five axioms of communication defined in 1921 by Paul Watzlawick based on the truth that humans communicate as soon as they perceive each other. This also extends to machines. Every user that comes to your website has already provided a flood of utterances. And the more they interact (or fail to interact), the

[5] Park, Y., JungHyen, A. Methodology for Automatic Ontology Generation Using Database Schema Information, 2018 *Advances in Mobile Networking for IoT Leading the 4th Industrial Revolution*

more we know about them from a personalization perspective. When you enter a store and say nothing, but have a confused look on your face, even the clerk knows to ask, "*can I help you*?" and based on other *signals*, such as what you are wearing and from where you entered, you've already started a powerful conversation before saying a single word, but again I digress.

By now you might be asking, *OK, sounds impressive, but make it more concrete for me.* That's more than a fair question that vexes many because there still exists a paucity of information on the subject. Here is how it can be done. First, you need a cognitive service. We'll use IBM Watson in our examples here. Take your encoded taxonomies, ontologies, and knowledge graphs (more about knowledge graphs shortly) and use a tool such as IBM Watson Studio™ to ingest them and create custom models.

The cognitive engine, using a facility such as IBM Watson Discovery™, uses all that intelligence to mine the content and perform content retrieval from a variety of content stores. With this method, terabytes of data can be reduced to only a few gigabytes of relevant data allowing more precise and effective semantic search regardless of the size of the corpus.

Building a solution using a set of cognitive services such as those provided with IBM Watson can certainly do the job, they remain what I like to call a "roll-you-own" proposition. In a bit we'll discuss some off-the-shelf strategies as few organizations have the development resources to do that.

Now that we can distinguish the differences and roles of taxonomies and ontologies, we can introduce the third portion of the semantic retrieval triad – the knowledge graph.

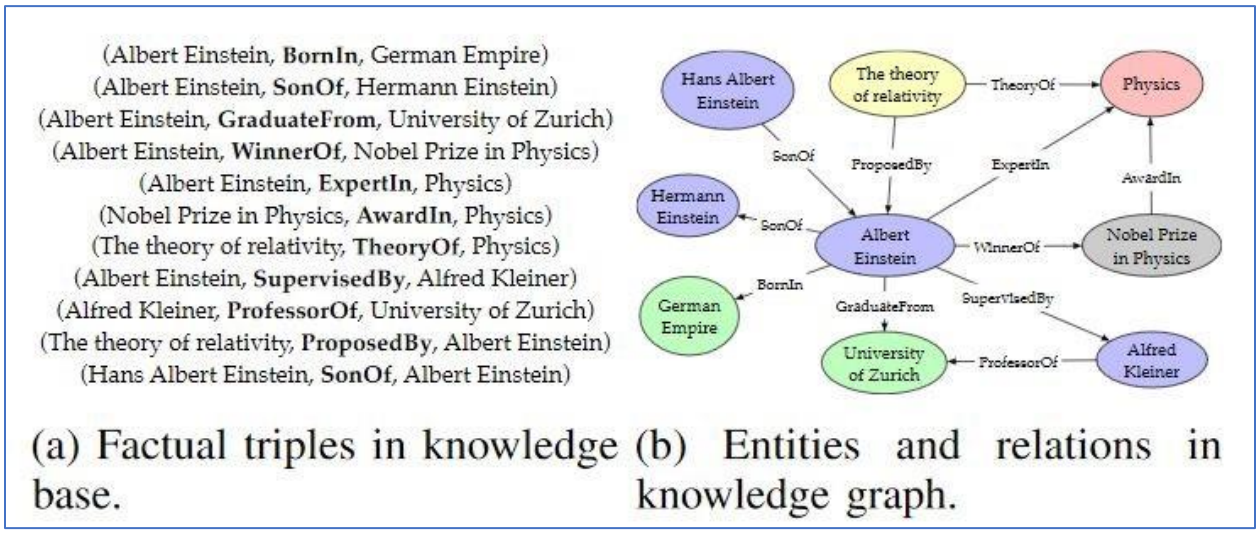
### Knowledge graphs

So, what's a knowledge graph? For the uninitiated, the word "graph" confuses many immediately. While a knowledge graph can be visualized, it's more of an inventory of relationships based on triples – the very same node model we use for creating and managing ontologies.

On the surface, a knowledge graph might seem nearly identical to an ontology that uses triples (subject + predicate + object relationships), but they go further.

An ontology sets the foundation for a knowledge graph to capture data; it serves as the backbone for a knowledge graph.

A knowledge graph is a visual representation of a knowledge base typically with *generic* concepts. It is a structured representation of facts, consisting of entities, relationships, and semantic descriptions. Entities can be real-world objects and abstract concepts, relationships represent the relation between entities, and semantic descriptions of entities and their relationships contain types and properties with a well-defined meaning.[6]




(a) Factual triples in knowledge base.

(b) Entities and relations in knowledge graph.

If they are so similar, why not skip the ontology and just use a knowledge graph? Well, some systems, such as PoolParty, integrate these.

The critical difference between the two is that ontologies provide a *concrete* representation of objects and their relationships. A knowledge graph on the other hand extends an ontology and makes extended *inferences* and *predictions* where an ontology alone cannot. The knowledge graph provides *context* and additional edge knowledge such as temporal and other information from which to draw those inferences. Knowledge graphs are relatively new in the past decade or so and are often represented using Simple Knowledge Organization System (SKOS and SKOS-XL), and the Resource Description Framework (RDF). The resulting databases provide the needed performance to process massive collections and

[6] Ji, S., Pan, S., Cambria, E., Marttinen, P., and Yu, P. S., "A Survey on Knowledge Graphs: Representation, Acquisition and Applications", *arXiv e-prints*, 2020.

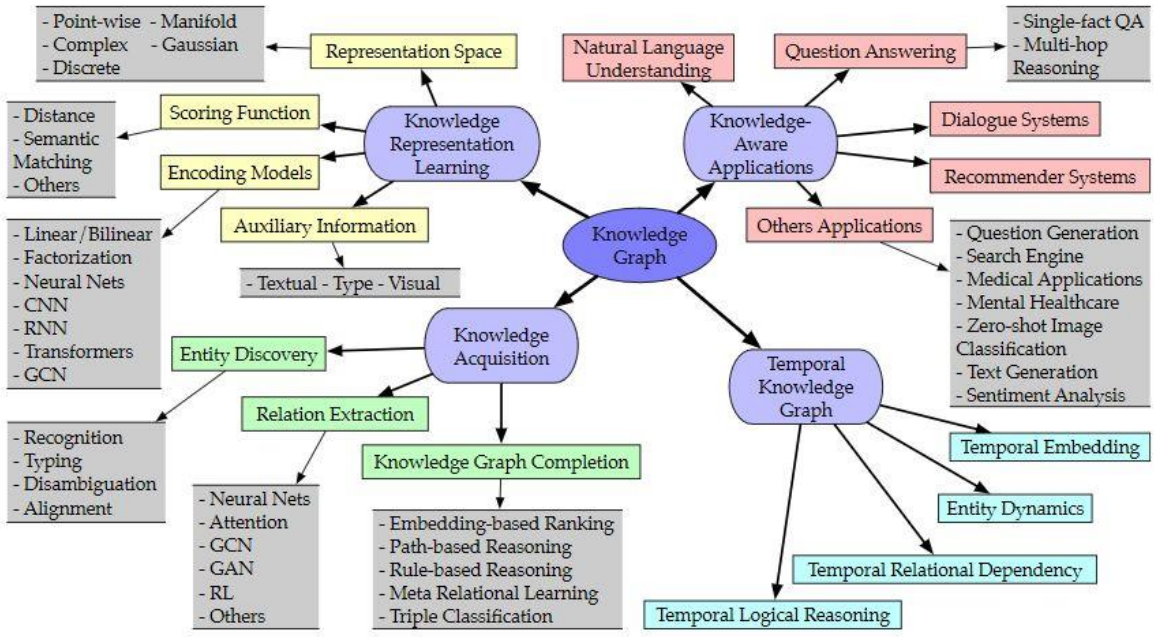
can create new knowledge from already existing facts using their inferencing capabilities.

Another way to think about the relationship between an ontology and a knowledge graph is that an ontology specifies the formal semantics of the data whereas the knowledge graph captures additional intelligence over the stored data. If that isn't clear as mud, one more way to think about the difference is that the ontology is the model, and the knowledge graph is the instance specific to your content domain.

A knowledge graph can be used to construct knowledge through inferred relationships and interact with an ontology that is then used to retrieve that knowledge. There are several number methods available for retrieving entities from knowledge graphs known as *Ad hoc Entity Retrieval from Knowledge Graphs* (ERKG). Knowledge graph information is stored in RDF triple stores and can be accessed using methods such a SPARQL Protocol and Recursive Query Language.[7]

There exists an entire ecosystem of research around knowledge graphs. The following figure[8] illustrates much of that related research.

[7] Kotov, Alexander. (2017). Knowledge Graph Entity Representation and Retrieval.
[8] Ji, S., Pan, S., Cambria, E., Marttinen, P., and Yu, P. S., "A Survey on Knowledge Graphs: Representation, Acquisition and Applications", *arXiv e-prints*, 2020.




## Scenarios graphs

Strongly typed content components using content typing architectures such as DITA provide a built-in classification scheme that semantically declares content purpose. For example, concept components are readily differentiated from task components because the intent is declared when the component is created. Therefore, strong content typing is essential.

Such differentiation of content types and the declared relationships between them at the source level enable retrieval and delivery of the right type of content for discrete purposes.

As an example, let's assume we want to ask our new cognitive content system the steps to do a specific task, such as find a specific recipe. No sweat. Our system discovers and retrieves the specific content object that contains the steps to complete the desired task and additional related topics. We might have discovered and delivered the same content using far less sophisticated methods.

But let's take it much further. Let's say we want to plan an entire event, a wedding on the beach in the Fall with a subsequent reception for 200 guests with catering and flowers. Now we need a whole lot more information from many sources. It's no longer a single task or precise answer – it's an orchestrated

scenario made up of dozens of hierarchical and parallel tasks. Can your search engine do that? I didn't think so.

This is exactly the problem many business organizations want to solve. They want to provide self-service content for complex scenarios using an array of product offerings. Users want to be *guided* through these complex scenarios based on *their* goals.

We need to tool our content systems with the ability to assemble *arrays of knowledge*. Cognitive systems based on structured intelligence can learn and suggest these complex patterns.

Now we are getting into some truly fuzzy areas that require more research and invention. Can we use knowledge graphs and cognitive systems to do this and do it well? Can we train the cognitive systems to do it? Why not?

A good content model includes a *content use model*. A content use model includes modeling of scenarios, not only discrete tasks. A yet to be explored question remains – can we encode pre-defined scenarios and use them as patterns to seed the initial training of the cognitive system above the knowledge graph layer? I believe so.

Many of you know me as a strong advocate and evangelist of DITA. I purposely avoided tying this entire discussion specifically to DITA because it can be done entirely without DITA., but without doing so it becomes that much more difficult. As I said early on, we're after creating and using intelligent objects. DITA just happens to be the best content object model in my opinion.

A core feature of DITA is its DOM organizational structure and inheritance model. We collect elements into typed topics, topics into related and purposed sub-collections, and sub-collections into purposed collections, and so on. As a result. these collections, called DITA Maps, are natural organizers for scenario *patterns*. We should consider using DITA Maps to abstract scenario patterns for consumption by a cognitive service and have the cognitive service learn and extend those patterns based on use. How might we do that?

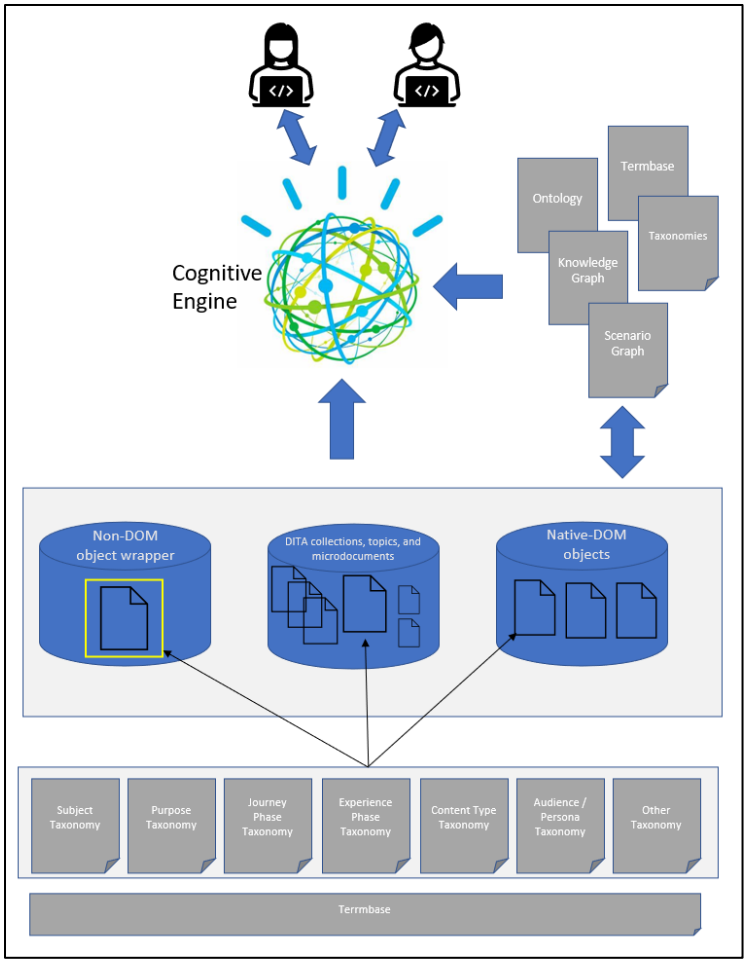
DITA provides a powerful build-in model to represent relationships between topics called *relationship tables*. Relationship tables are normally used to add links

and navigation between topics using *indirection*. Relationship tables help make topics (which are also encapsulated objects) portable.

Let's consider that for a moment. With relationship tables, we're overtly declaring relationships between objects that can be used when constructing a knowledge base. If we wrap non-DITA content objects with DITA and use it to model scenarios using relationship tables, can we encode them to automate the construction of the graphs? How might we do that with tools? We might consider using a combination of tools such as an XML transformation tool and a search appliance.

Additional research is needed in this space. There exists little to no references on the notion of scenario graphs to model user assistance nor their use in the construction of related knowledge bases.

Let's examine a conceptual diagram of what our (admittedly over-simplified) cognitive content supply chain might now look like.




## Amplifying the signals with blockchain

Recall our discussion about utterances? Utterances are signals. There are at least two key aspects about utterances as they relate to our discussion about enabling a cognitive content supply chain:

1. Improving the signal-to-noise ratio.
2. Dynamically adjust content delivery in response to changing signals.

Again, we're not limiting our definition of what constitutes a signal. A signal is whatever form of utterance our systems are tooled to receive. Even lack of a signal *is* a signal, including pauses when we recall the basic axiom that we cannot *not* communicate.

Conventional thinking might have us limit what we collect as signals to the most obvious ones, but what if we could trace those signals based on what a user does

as opposed to only what they explicitly ask? How would we go about tracing the activities of a user - with their permission of course where privacy is concerned?

Blockchain is a powerful technology that at its very essence is an immutable ledger of transactions. Most associate blockchain with only Bitcoin and financial transactions. But what if we use it instead to capture the user actions, store them, and use elastic search to mine it and return new, amplified, multiple, and changing signals on the fly?  Armed with the right signals we can adjust the delivery of user assistance proactively. In doing so we've just crossed the once impassible divide between failure-mode user assistance and proactive user assistance. Taken further we might offer to invoke agents to perform tasks for the user based on what they are trying to accomplish.

### Actual use

We now have everything we need to help a cognitive service perform accurate content retrieval and assembly. The only piece of the puzzle that remains is to deliver the retuned content to the customer through the desired channel(s). There are no off-the-shelf offerings of which I am aware that directly interface to one of the major commercial cognitive systems in the marketplace – yet we can be sure they'll eventually come, or we can build our own now that we have the requisite collection of intelligent content assets. There are, however, commercial offerings we can implement as tactical solutions that use similar methods even if those solutions are not yet fully open. All the work we've done up to this point helps the function and precision of these systems as we prepare for even more advanced cognitive-enabled content delivery solutions.

### Tactical solutions

If all of this sounds big blue sky that's quite understandable. If you have attended any of the hundreds of webinars and conference sessions about intelligent content, content modeling, journey mapping, personalization, AI, and related subjects you'll now readily recognize this major gap – how intelligent content is ultimately retrieved and delivered that leverages all the powerful intelligence you've diligently added to your content asset – and do it across multiple content silos.

As I mentioned earlier, when you start asking tough questions about how ontologies and knowledge graphs factor into dynamic and highly personalized

content retrieval and delivery, you'll more often get blank states or nonsensical babble than good answers. If you receive a response that mentions triples and inferencing, that's someone with whom you want to continue the conversation.

As touched upon earlier, the sad truth is that even those in content leadership roles punt blindly to content systems providers. We need the content systems providers to enable their platforms with a choice of cognitive services and accept open standards for ingesting intelligent content assets such as taxonomies, ontologies, and knowledge graphs, synonym rings and other related assets. A few can.

Like early structured content systems, a cognitive supply chain is still a roll-your-own proposition, but there are select few content platforms moving in the right direction and are using (or mimicking) some of the mechanisms discussed in this paper, although they generally remain semi-closed and proprietary when it comes to their AI capabilities and interfaces.

As a sidebar allow me to give deference to the CMS providers, especially the cCMS providers and consultants. They need revenue to do the development we as content producers want and cannot afford to build ourselves. Yet there are many in our industry that focuses on only the lower half of the intelligent content equation - pushing mostly the efficiency aspects of the value proposition, which is real, but giving only lip-service to what organizations really want - true dynamic one-on-one automated personalization at scale with scenarios and solutions in both prescriptive and dynamic, non-prescriptive content assemblies.

Yet we *are* on the right track. None of the work we've done by defining content models, content journeys, terminology, taxonomies, harmonization, and more is going to waste – it all improves traditional content search, retrieval, and personalization. It is worth staying the course.

So, what can we do in the meantime?

I am personally averse to all-in-one content platforms. Think about it critically. Technology for creating and managing content is typically optimized for domain-specific purposes. Technology for the kind of advanced content retrieval, personalization, and delivery we are after is a different game altogether. I strongly advocate for the separation of content creation from content delivery when

selecting content platform components. The problem with all-in-one solutions is that you end-up taking the not-so-good parts with the good ones and often have to force-fit different types of content with one another, such as text-heavy technical documents with graphically intensive marketing content.

There's a class of content management systems that called *headless CMSs*. Although they might provide a customizable portal, the use of the front-end portal is optional of you have one or more existing delivery front ends. These platforms enable you to intentionally separate the source content management back-end from front-end content delivery. These are the ones I prefer in the race to cognitive content.

Used properly, some can act as a type of *content middleware* that can do both omnichannel content sourcing independent of performing omnichannel content delivery. They can be used as a canonical content aggregator, assembler, and broker that is truly channel-agnostic and independent. I call it the *Content Cloud*. The content cloud sits between multiple content management systems on the back end, and multiple delivery channels on the front end. Some of these systems already use some of the concepts described in this paper even though they may not yet be fully there or open yet - they are on the right path and you can use them right now to aggregate content from multiple content silos and deliver that content to any channel. The best of these can ingest and use your intelligent content assets, such as an OWL-based knowledge graph.

It is up to us, the content owners, to demand more from the supplies to enable their platforms with cognitive services and intelligent content assets.

## Summary

Let's not lose sight of the basic premise of this model. It employs a cognitive engine that continuously improves long after it is put into use; our system learns. That is the key differentiator between this model and conventional content management systems. As mentioned earlier there are a few component content management systems that can ingest and use the intelligent content assets described herein including controlled vocabularies, ontologies, knowledge graphs, and more. These solutions provide a reasonable tactical option if you cannot build

a cognitive-enabled content retrieval and delivery platform yourself, but unless there exists a cognitive engine behind the CMS, the platform cannot learn and improve with use – it isn't a true AI-based cognitive system.

What we're doing with this approach is hyper-accelerating the training and precision of a cognitive content management system. Taxonomies, ontologies, and knowledge graphs provide the essential intelligence that cognitive engines need. The core premise of this model is that it is better to start with intelligent (enriched) structured source content. Many are forced to start with only unstructured content. This model leverages a structured-to-structured approach to its fullest.

Earlier there was also a brief mention of was called autonomic content. Autonomic content is self-reconfiguring and self-healing content. It goes beyond most existing cognitive models. If we have a structure-in and structure-out model with our intelligent assets, then the system might can be designed to support continuous and automatic bi-directional improvement that tunes and corrects the content as the system learns based on use.

## About the Author

Michael Iantosca is the Senior Director of Content Platforms at Avalara Inc. He has spent the entirety of his 40-year career as a devout content professional and pioneer.

Joining Big Blue in the Fall of ’81, he spent thirty-eight years at IBM leading the design and development of advanced content management systems and technology that began at the very dawn of the structured content revolution. It was also the dawn of practical hypertext, long before the emergence of the Mosaic web browser and the world wide web. He began his career at IBM’s Mid-Hudson Valley lab developing Hollywood-class, 40-foot-wide rear-projection multimedia extravaganzas for large corporate events. Those productions used dozens of sequenced projectors that were programmed using an AVL microcomputer before the first IBM personal computer rolled off the manufacturing line.

As an IBM Information Developer, Michael was also trained by IBM as a systems software engineer while attending graduate studying computer science at Marist College where he received an undergraduate degree in Communications with concentrations in journalism, broadcasting, and psychology.

Michael holds the earliest published invention disclosure[9] for embedding hypertext-driven media players in electronic books using structured mark-up. That effort connected him with the inventor of GML and SGML who had been asked to review Michael’s invention disclosure, after which Michael was asked to lead the design and development of IBM’s first SGML-based publishing platform. He subsequently led the migration of millions of pages of IBM content to SGML and later, DITA XML in the 90s.

Michael was also responsible for forming the XML team and a member of the working group at IBM that invented DITA. If Michael hadn’t prevailed in a pitched internal battle to develop an XML platform over a planned SGML variant called WebDoc in the mid-90s at Big Blue, DITA, and the entire industry that supports it, might not presently exist. He went on to build the first DITA-based publishing system and subsequent generations of those systems and holds several DITA and AI-related patents and invention disclosures.

Michael is an avid digital and large format film photographer going on fifty years and maintains a formal portrait studio and traditional film darkroom in additional to advanced digital imaging. He lives with his wife Ellen overlooking the majestic Blue Ridge Mountains near Asheville in Western North Carolina.

[9] Iantosca, Michael J. Invention Disclosure, July 1992, Fully Digital GML Based Authoring and Delivery System for Hypermedia, PO891-0201, IBM Technical Disclosure Bulletin Volume 35 No. 2

# References


Ji, S., Pan, S., Cambria, E., Marttinen, P., and Yu, P. S., "A Survey on Knowledge Graphs: Representation, Acquisition and Applications", *arXiv e-prints*, 2020.

Blumauer, A., Nagy, H. "The Knowledge Graph Cookbook: Recipes That Work", *monochrom*, 2020, *Semantic Web Company*, 2020.

Manhaes, M., Ko, T., Selim, A., Amer, 0., Sri, L. Building Cognitive Applications with IBM Watson Services: Volume 4 Natural Language Classifier, 1985 IBM Redbooks

George, Palliyathu Vishal (Babusapalya, IN), Iantosca, Michael J. (Wake Forest, NC, US), Kurian, John (Bangalore, IN), Sankar, Balaji (Bangalore, IN), 2019 UNSTRUCTURED DOCUMENT MIGRATION, United States INTERNATIONAL BUSINESS MACHINES CORPORATION (ARMONK, NY, US) 20190205460

Iantosca, Michael J. (Wake Forest, NC, US), Jenkins, Jana H. (Raleigh, NC, US) 2014 LINGUISTICAL ANALYTIC CONSOLIDATION FOR MOBILE CONTENT, United States, INTERNATIONAL BUSINESS MACHINES CORPORATION (Armonk, NY, US) 20140088953

Iantosca, Michael J. Invention Disclosure, July 1992, Fully Digital GML-Based Authoring and Delivery System for Hypermedia, PO891-0201, IBM Technical Disclosure Bulletin Volume 35 No. 2

Kotov, Alexander. (2017). Knowledge Graph Entity Representation and Retrieval.

Park, Y., JungHyen, A. Methodology for Automatic Ontology Generation Using Database Schema Information, 2018 *Advances in Mobile Networking for IoT Leading the 4th Industrial Revolution*


# Trademarks