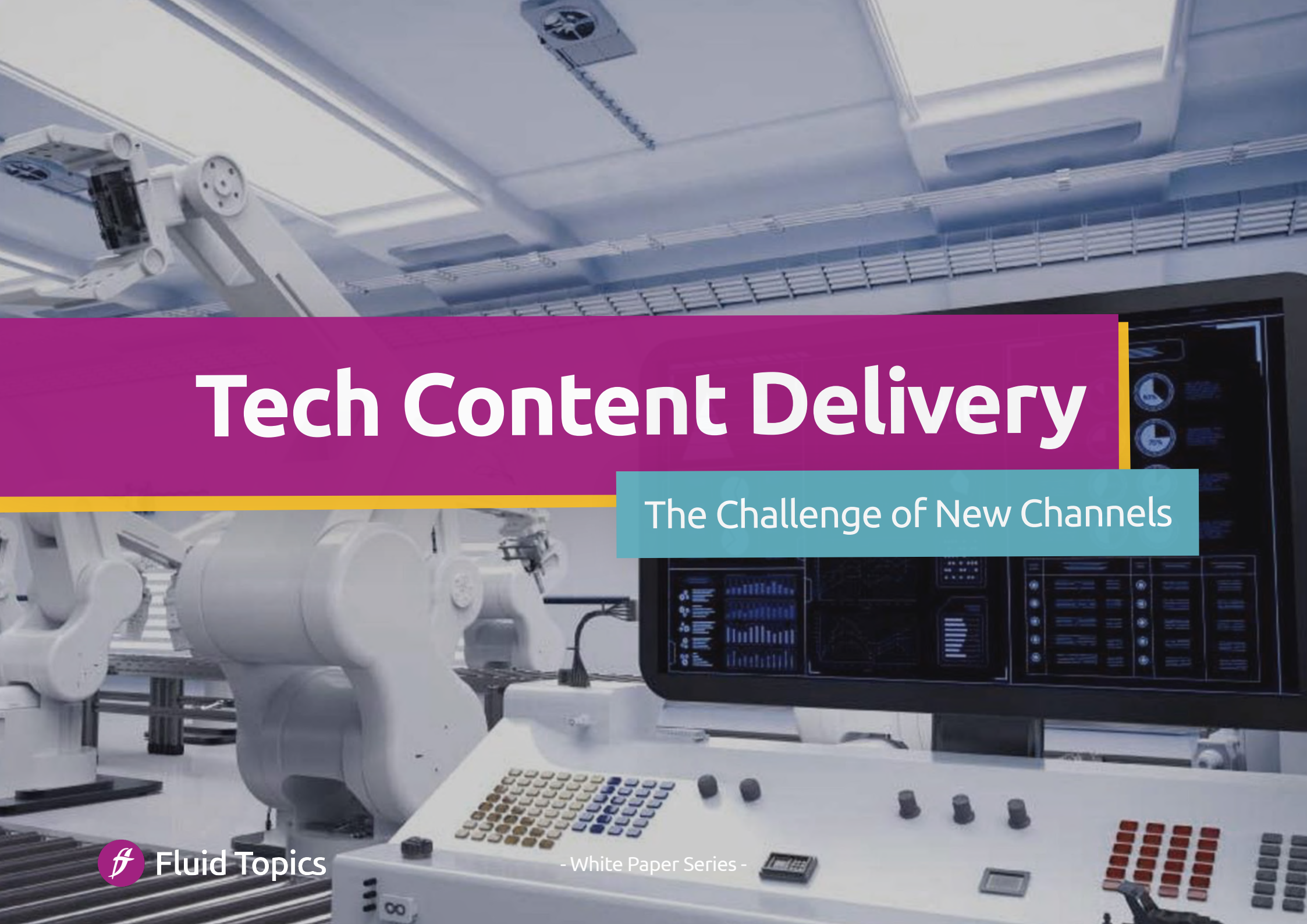


Tech Content Delivery

The Challenge of New Channels

Fluid Topics

- White Paper Series -

The New Frontier of Content Delivery

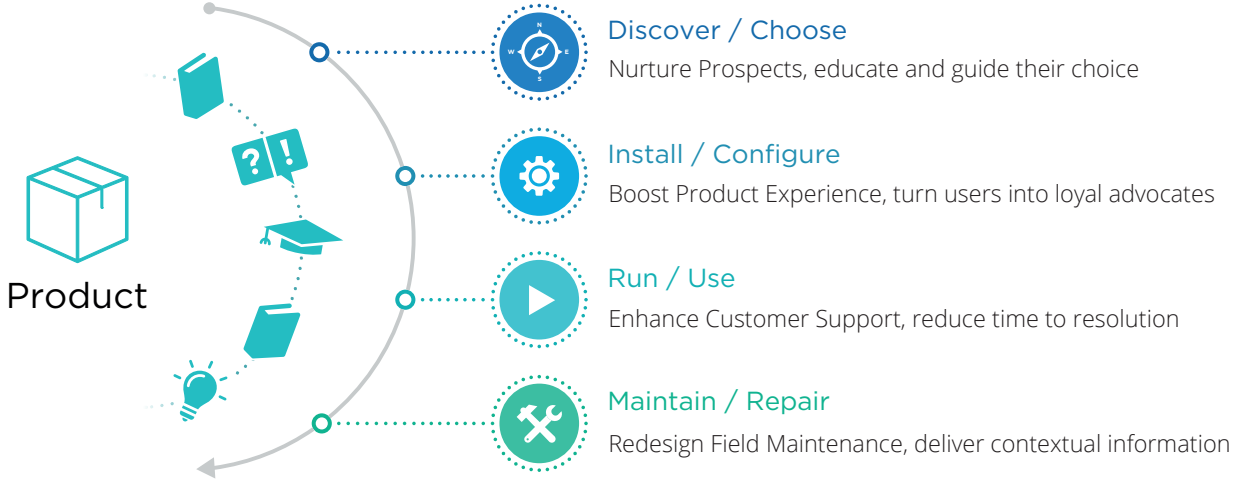
Technical documentation is a part of the experience we have with our products. It is essential to many phases of the customer journey, from pre-sales, installation, use and maintenance through to support. Historically delivered on paper in the form of books or manuals, technical documentation is a narrative provided via words and phrases, as well as images, written by humans to be read by other humans.

The advent of digital technology has disrupted this state of affairs, raising all sorts of questions concerning information architecture. The former supremacy of large books and manuals has been upended by the emergence of short content focused on individual topics, such as articles and knowledge base entries. This has engendered fresh challenges on how to organize this torrent of information in order to make it consistent and meaningful. Embodied by products such as Fluid Topics, Dynamic Content Delivery has managed to provide an effective response to this transformation.

The shift from book to article remains firmly within the tradition of content that is written and read by humans in the form of typed text, but it falls short in accommodating the next step. Indeed, a revolution is under way, one that will introduce exciting new tools for accessing and using information; readers will see new interfaces with fresh and different user experiences, far removed from traditional pages of text. This revolution combines:

- Self-documented devices that give predictive and contextual indications.
- Augmented Reality, already present on tablets and mobile devices.
- Wearable tools such as smart glasses.
- Interactive agents, already standard fare with Apple's Siri or Amazon's Alexa.

Technical content is essential to product adoption and customer success:



The New Frontier of Content Delivery

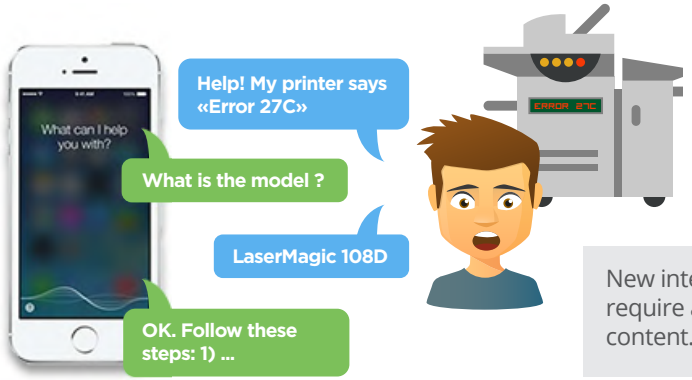
These innovative techniques for consuming content sound appealing, but they require us to rethink our approach to technical communication. Can anyone seriously imagine reading a three-page PDF article using connected eyewear? Or listening to an intelligent agent reading a ten-minute response to a question? Technical documentation as customarily produced, and especially as it continues to be published today in the form of short or long texts, is clearly not up to the evolving nature of the task.

Legacy documents are clearly unsuited to advanced devices.

Information professionals must take stock of the challenges and prepare for them. They must write for machines; in other words, produce a form of information which consumption and use are in line with these new tools.

While it is easy to scan a book, it is hard to teach a computer to answer questions about its contents. The computer is badly prepared to be a reader, and books are badly suited to be read by machines. It may be possible, though, to prepare text in such a way, and program our devices in such a way, that they can present the content to us **as if they understood it**.

To illustrate what needs to be done, we'll consider one channel that is getting attention lately, the chatbot. One use of the chatbot is in the digital voice assistant, which, according to Juniper Research[1], will reach over half of US households by 2022. This technology is taking off as a platform for content consumption. Adapting content to work with chatbots is a challenge.



New interaction channels require another form of content.

1. https://www.juniperresearch.com/researchstore/innovation-disruption/digital-voice-assistants/platforms-revenues-opportunities

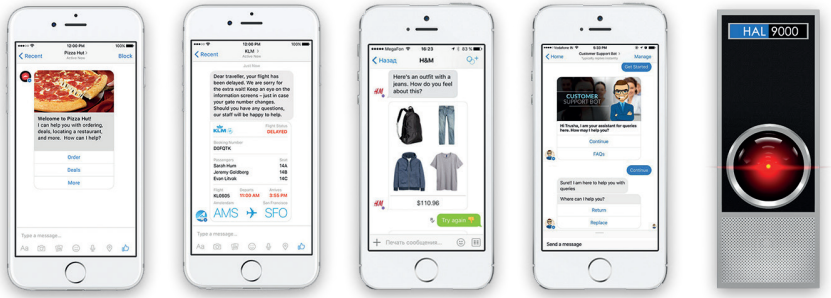
The Rise of Chatbots

Chatbots and Virtual Agents are very much in fashion: everybody needs to have one. Your company doesn't have a chatbot strategy or even a project? FoMO!

Even though it has been a long-time dream and research subject for decades, the current wave of interest started with Apple's Siri in 2011. Siri had limited capabilities at the beginning, which made it the butt of many jokes, but it improved rapidly. Then came Amazon Alexa (2014) and Google Home (2016), with their physical embodiment through home devices.

It's interesting that this technology first emerged through the consumer market. Maybe because the complexity is daunting and pure B2B markets were too small to trigger the investment. It's also worth noting that there was no particular demand for such products. It's because foundational technologies reached a certain level of accuracy and performance (mostly due to the recent progress of machine learning) that it allowed web giants to develop these products, educate the market and anticipate how they can blend into your day-to-day life, not to mention reach for your wallet.

So, the first use cases were B2C, with thousands of very focused and limited applications. Well-known examples are ordering pizza, eCommerce, or social media. The reason is simple: the narrower the scope, the easier it is to be accurate. These simple conversational agents were called chatbots. With the evolution of technology, we see more sophisticated use cases and the rise of Virtual Assistants. As opposed to chatbots that are very specialized, virtual assistants are supposed to have broader capabilities and be used for more open-ended tasks: customer support, employee assistance, and more. We see this with Siri, whose skills have grown significantly since 2011.



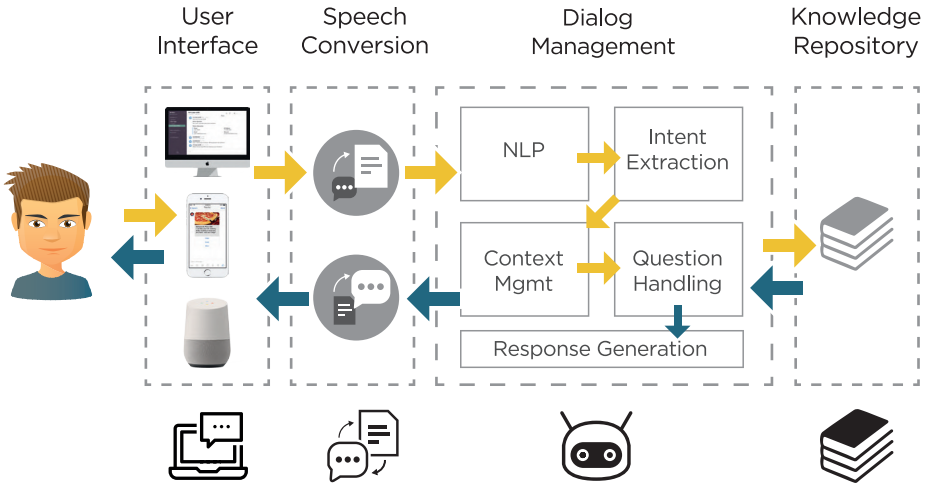
Pizza Ordering | Travel Booking | Shopping | Customer Support | Astronauts' Entertainment

So, as the technology matures and the cost goes down, we see many B2B business cases where it could be applied. Dozens of tech vendors have emerged trying to build their offering, and studies show that more and more companies are running pilot projects to evaluate the potential. Analyst firm Gartner predicts that by 2020, 25% (up from 10% in 2015) of customer service and support operations will integrate Virtual Agents technology across engagement channels. But to really understand what use cases can be accomplished today and where AI-based robots can be used in your company, it's necessary to look at how they work. We will then grasp why connecting a generic chatbot to a body of knowledge made up of a large set of textual content (DITA, HTML, PDF...) through a simple search engine can't work. Beyond a carefully staged demo presentation that makes it look so easy, there is a genuine complexity that requires new approaches and new advances.

Architecture of a Virtual Agent

A chatbot is a computer program designed to simulate human behavior and to interact in a natural way. The tricky part of this is "natural way", which implies holding a conversation, either text-based or verbal, through a variety of channels.

Let's review the different layers at play:



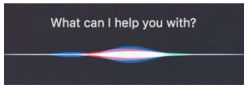
User Interface

The ways and devices you can use for connecting and interacting with a chatbot are diverse:

- A search bar or a chat window in a web browser
- A messaging application (Facebook Messenger, WhatsApp, Slack, MS Teams...)
- A business mobile app having conversational capabilities for specific purpose
- Embedded in the operating system (Siri is now part of iOS and MacOS)
- Dedicated hardware devices (Alexa, Google Home)

The interface is not the main issue. Thanks to the API-first approach of all modern applications and the different frameworks available, it has become fairly straightforward to implement a connection between any type of UI and the next layer.

Mobile App
(dedicated or embedded in)



Operating System

Social Network
(open & public or controlled for business)

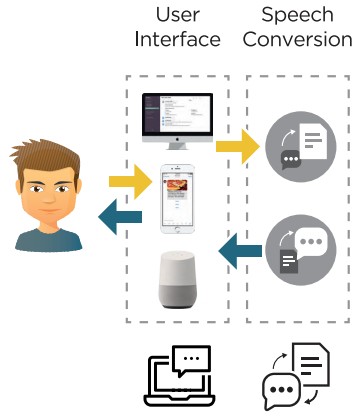
Appliance
(home or professional)

Virtual agents used to be separate visible apps. But they become more and more pervasive and embedded into existing artefacts.

Architecture of a Virtual Agent

Text vs. Voice Interaction

Some of the communication channels mentioned above are voice based. Let's get past the verbal aspect because it is, after all, just an optional intermediate layer, and it will always be supported by text.



Speech is amazingly complex due to many factors: noisy environments, accents, emotion... And everything needs to sound natural or users will reject it.

These two directions (Text-To-Speech and Speech-To-Text) are very different challenges, but somehow interconnected. Without digging into nerdy technology, we can at least state that they have two things in common:

1. Forget the crude old "robot voices" we used to hear on science fiction shows.
2. The natural speech problem has been solved quite recently (2017) thanks to Deep Learning.
   - Speech recognition now reaches about 5% to 5.5% error rate, which is surprisingly close to human capabilities (about 5.1%).
   - Voice synthesis with WaveNet, the application of deep neural networks model to raw audio waveform generation, has been amazingly successful. Since December 2017, it is almost impossible to distinguish human from AI generated voices. You can find many impressive examples and demos on the web[1].

1. https://google.github.io/tacotron/publications/tacotron2/index.html

Therefore, speech is based on text, and we have boiled the problem down to a text-based interaction. Thus, the two remaining core parts of a chatbot to discuss are:

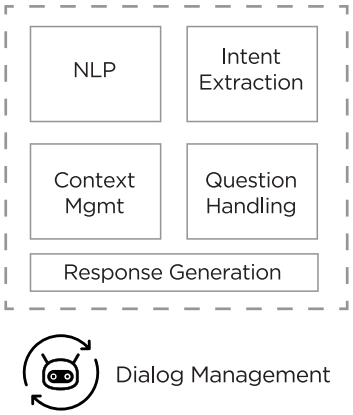
1. Dialog management
2. Question answering

Dialog Management

This layer is in charge of parsing and analyzing the text it receives to:

- Manage a coherent dialog. It is the part of the bot in charge of small-talk, such as greetings and standard corporate boilerplate.
- "Understand" the intent of the user – the "what are you talking about" part.

This involves multiple sub-modules and technologies:

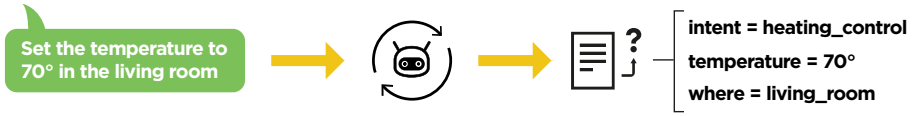


- **NLP** (Natural Language Processing) for language detection, part-of-speech tagging (grammatical analysis), semantic analysis, entity extraction... with general and domain specific understanding.
- **Context management**: to grasp the context, the chatbot needs to remember previous interactions; dialog can (and usually does) span multiple interactions (phrases), during one or several sessions spread in time. It must also be able to integrate user preferences, previous customer purchases, etc.
- **Intent extraction**: based on NLP and context, the chatbot must determine what the user expects. It must create a synthetic representation of the dialog — the utterance — that can be forwarded to the next layer.

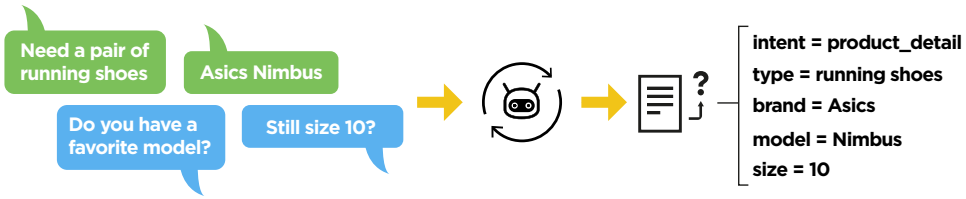
Architecture of a Virtual Agent

Examples of intent extraction:

- From a simple utterance



- From a dialog spanning over multiple phrases



There are dozens of frameworks intended to do this: Cloud-based or on-premises, from ISVs or Open Source, specific or generic ...

To name a few:

- From web players: Amazon Lex, Facebook, Google Assistant, Microsoft Luis (Language Understanding Intelligent Service), IBM Watson...
- From ISV or Open Source: Aspect, ChatterBot, Dialogflow, Nuance, Pandorabots, Rasa, Semantic Machines, Wit.ai...

Choosing a framework is difficult and depends on multiple criteria: language support, domain specialization, privacy, offline/on-device, pricing, etc. The landscape is evolving rapidly, so projects have to be tactical and must be prepared to evolve quickly.

Answering Users' Questions

Let's assume that we have chosen and properly set up the dialog management layer, and that for each user dialog we have an intent properly extracted. Now starts the tricky part: generate a reply.

What a chatbot can actually achieve is driven by what it is connected to.

If it needs to assist a human, answer questions, or take action, it has to have some knowledge. Which leads to a simple question: how is this knowledge made available to the bot?

We can distinguish three cases:

1. The knowledge is a **structured source** (such as database) accessible directly or through APIs.

2. The knowledge can be modeled in a **decision tree** or a small knowledge base with short textual records.

3. The knowledge is diluted in a **large body of textual content** (articles, books, web pages ...) originally written for humans, and stored in one or multiple sources.

Let's review the possibilities and strategies for each of these situations.

Knowledge formats

1st case: the knowledge is a structured source

Today most of the chatbots work following a rule-based model:

```
Ask question Q. If reply is X then say Y and ask Z.
```

This approach requires the pre-definition and scripting of all possible cases and interactions, to model dialogs for all subjects you want to support,

This works well for simple use cases such as pizza ordering: you have 3 sizes, 10 toppings, and 2 types of crust. So, asking the questions and getting the answer by matching them to a known list of ingredients isn't rocket science. It can be tedious to build, but it can be done.

This is also easy for e-commerce where a few rules can be created to ask for a type of product, brand, color, size... and to match the answers against the different fields of the product catalog.

To illustrate this, let's consider this simple question:

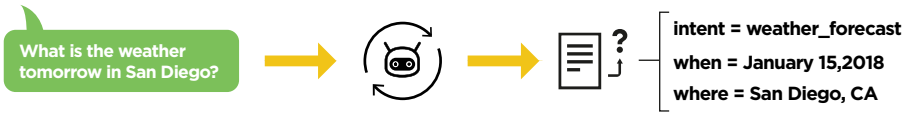
"What is the weather tomorrow in San Diego?"

It is parsed by the dialog management layer and transformed into a structured intent query:

Intent= weather_forecast

When = January 15, 2018

Place = San Diego, CA



And you have access to a database that contains records providing the rainfall, temperature, wind speed and direction, per day and hour, past or forecasted for the next 10 days:

Country	City	Date	Time	Pluvio	Temp	Wind Speed	Wind Dir
USA	San Diego	2018-01-15	12 pm	0mm	15°C	14km/h	270°
USA	San Diego	2018-01-15	2 pm	0mm	17°C	18km/h	260°

It looks pretty straightforward to:

1. Transform the user's question into a database query, such as:

```
Select P=sum(Pluvio), T=avg(Temp), W=max(Wind_Speed) from weather_forecast
Where City='San Diego' and Date='2018-01-15' and (time >=6am and time<= 10pm)
```

One can easily see how a SQL template can be filled in with the elements provided by the intent.

2. Get the result, and fill in a reply template with the value.
3. And even to include simple rules such as:

```
if (chance_of_rain > 30%) then say, "You might need a raincoat".
```

We can conclude that if the knowledge repository is a structured source, the different fields have strong semantics. This helps avoid ambiguity, which makes it easy to define rules for mapping user requests into formal queries. The more databases you have, the more types of question (i.e. intents) you can handle. This is how Siri and other Personal Assistant extend their skills, connecting to more sources: catalogs, traffic, your calendar, and many more.

Knowledge formats

2nd case: the knowledge can be modeled in a decision tree or a set of Q&A

The back end is a database of textual knowledge records designed specifically to answer user questions, and it contains enough metadata (tags) to help choose amongst the entries which one to use based on the context.

This Knowledge Base (KB) could have been created for support agents in a call center, for instance. Say that we are a telecom operator and we have a KB for answering questions such as, "*How do I change my plan*", "*How do I activate Voicemail*", "*What are the international roaming charges*" ...

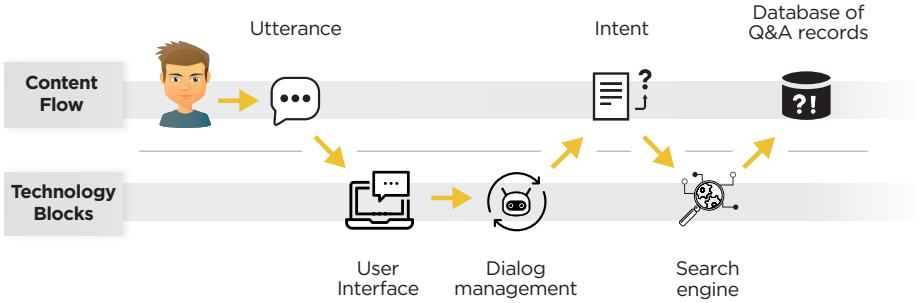
In this case, we can either:

- Transform these KB entries into rules according to a format supported by the chatbot framework such as AIML[1] and load them into the chatbot.
- Keep the content external in the database and use a search engine to query and retrieve replies based on the intent.

1.https://en.wikipedia.org/wiki/AIML

The former approach requires more work and understanding of information design but should give more controlled results.

The latter way allows a simpler and more scalable approach (particularly if your KB has thousands of entries) but could give less accurate results. To mitigate that loss of accuracy, we could define dialog rules to gather context from the user in order to generate a more targeted search query. So, in the case of the question about roaming, the bot could ask which country the customer is visiting. If an initial question leads to multiple possible answers, maybe asking the device type of the user and turning that info into a filter in the search query could reduce the number of results and lead to the right answer.



3rd case: the knowledge is diluted in a large body of textual content

This knowledge is possibly in different formats and comes from multiple sources. This is the most interesting and the most complex scenario. It is also the one that corresponds to technical documentation.

Chatbots: Beyond the Demo Effect

Let's start by looking at a typical demo or proof of concept you might see. Even though it's totally made up, in some ways it contains the ingredients of the solution.

1. The UI front end is the Slack messaging app, which fits well in a B2B environment, and because it's usually natively supported by most chatbot frameworks. After a rapid smalltalk interaction (also natively handled by the bot but always a good way to start to impress), the demonstrator types a simple funny question such as 'How do I cook spaghetti'.
2. As explained before, the chatbot framework transforms this simple utterance into an intent. The intent is mapped into a search query, and the query is run on the content, just as you would do manually with keywords and facets. The only difference here is that the keywords and the filters have been inferred by the bot layer from the dialog.
3. The first result of the list returned by the search engine is presented to the user as a reply.

The reality behind the curtain is that to have an effective demo, the indexed content must be a small set of topics, carefully designed for that purpose (usually an intro phrase plus two or three steps of actions), and the words used in the dialog are carefully chosen to match the text of the topics. Without too much work (a few days) you can have an impressive demo.

But this is misleading, as it doesn't apply to the real world:

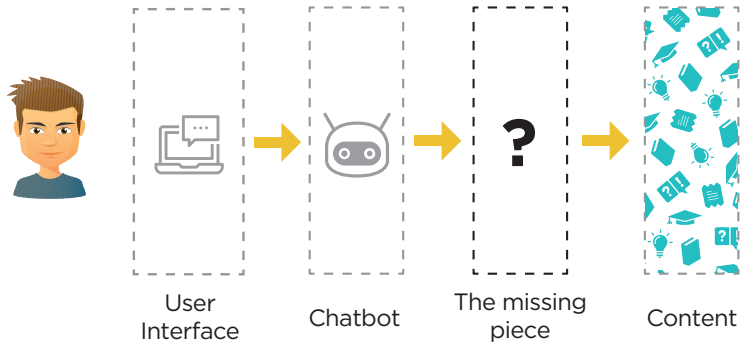
- On the content side: Documentation is usually not made of small fragments designed for one particular use case. At best, you write structured topics (using formats such as DITA), but these topics are designed for reuse, or for being self-sufficient (sections, chapters). At worst, you have large documents (tens of pages) or long web pages. Anyway, there is a very low probability that your content is engineered to have the appropriate granularity so that the fragments can be displayed separately from one another in a chat window, to reply to a user question.
- On the technology side: the search engine not only has to be semantic (to find replies even though the user's words are not explicitly in the content), but also highly relevant. If the strategy of the bot is to display the first result of the returned list, you have to ensure that this is the right one. Having a good reply to a search query on the first page (among the 10 or 20 top results) is already a daunting challenge. So, assuming that, using the same technology, you can blindly take the first result and return it as the reply seems doomed to fail.

Does this mean that it is impossible to connect chatbots to a vast body of textual knowledge? No, we think there is a solution. But it requires a less naïve approach and a deeper take on the technology.

Connecting to textual content

Let's first restate the assumptions of a real use case:

- The content is almost exclusively textual with images, whether authored in structured XML (such as DITA), or totally unstructured: Web pages, Wikis, long knowledge-base articles, or legacy documents such as PDF or Word. You usually have more than one source, which brings more opportunity to find the answer to the question, but also brings more complexity due to the variety of formats and the lack of consistency (metadata, size, etc.).
- We will rely on a conversational framework that handles the upper layer (connection to the channel, chit-chat, context management, intent extraction).



Under these assumptions, our problem can be reframed as this simple question: **How can we connect the framework to the content?** What is the missing piece? Is it just technology? Does it require us to redesign our content?

Let's first dispose of the "rewrite your content and transform it into rules" approach. It doesn't make sense to transform documents into a rule-based expert system, because:

- It is impossible to design a formal model that can represent human knowledge, even in a limited domain.
- It is economically insane to rewrite the knowledge conveyed in thousands of pages into rules. The profiles and expertise required are not the same as these of tech writers and would require AI experts.
- Maintaining the model over time is even more complex than writing it in first place.

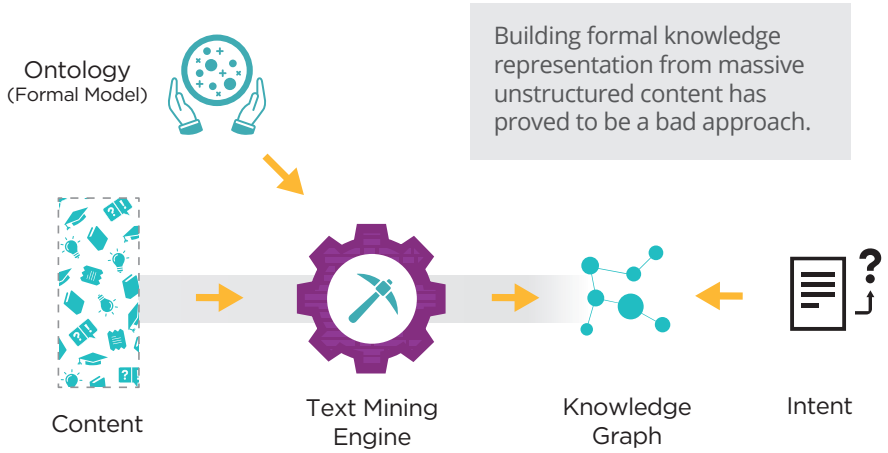
This approach was tested in the 80s and has shown its limits[1]. As summed up in Wikipedia[2]: "The most common disadvantage cited for expert systems in the academic literature is the knowledge acquisition problem."

We can also immediately discard the naïve approach of aiming a search engine directly at the content. As we have explained, the content itself is not suited (too short, too long…) and the generally available search technology is not good enough.

1. https://www.jstor.org/stable/2582480
2. https://en.wikipedia.org/wiki/Expert_system

Connecting to textual content

Then, let's look at a school of thought that could be called the "Information Extraction". It consists of applying various text-mining technologies to transform the textual source content into a formal structured representation of the knowledge conveyed by the text, and then to query this constructed knowledge base instead of the original text.



The word that describes this approach best is Ontology. Wikipedia defines an ontology[1] as "a formal naming and definition of the types, properties, and interrelationships of the entities that really exist in a particular domain of discourse."

It means that you need to define all the knowledge objects (in other words, concepts) that you want to represent, their properties (or fields) and their relationships. If you are more familiar with the classical relational database world, see that as defining all possible tables, columns, types, joins... the entire model. The difference being that, instead of relational models, ontologies rely on graph technology (OWL, RDF...), which is why the result is called a Knowledge Graph.

So once the model is defined, you need to set up text-mining technology that would read the content and automatically populate the graph (or fill in the tables).

One can easily foresee the problems:

- Building the ontology is the first challenge. This is not only because of the skills required, but because representing knowledge is exponentially more complex as the richness of the model grows. The more concepts you want to include, the worse it gets.
- Automatically extracting knowledge from the text to populate the graph (or the tables) with enough precision (i.e. don't fill it with the wrong data) and recall (i.e. don't miss anything) has proven difficult. And the errors that are inserted at that phase get amplified when the system is queried, which makes it useless.
- Transforming a user query into a graph query is also very complex.

This approach has been the focus of research and investments in the past 20 years. It has been shown to be valuable in specific use cases, but overall it has lost traction. The latest machine learning algorithms (using deep learning) have proven better at extracting info from the text, but designing a formal detailed representation of human knowledge is still a challenge.

1. https://en.wikipedia.org/wiki/Ontology_(information_science)

Connecting to textual content

Given all that we have discussed so far, there is a middle way that offers a viable approach. Instead of chasing the dream of mimicking the human brain, the path that has proven to be the most efficient right now that has the best ROI (quality versus complexity and cost) is **a mix of content enhancement and search technology**. In contrast with the previous approach, we could call it the school of "Information Retrieval"[1].

The name itself hints that it has to do with search technology, but in a new way that entails two main aspects:

- **Enhance content**, which means semanticize, or to use a less pedantic word, label the meaning of the words in a clear and obvious way.
- **Index** this enriched text with a new breed of search technology that permits it to transform user intents as provided by the upper framework into smart queries.

In the next chapter, we go into some detail on how to process text to make it renderable by mechanical intermediaries.

1. This approach, mixing enhanced content and advanced search technology, has proven successful. This is the paradigm used by the initial IBM Watson program for winning at Jeopardy! http://www.nytimes.com/2011/02/17/science/17jeopardy-watson.html

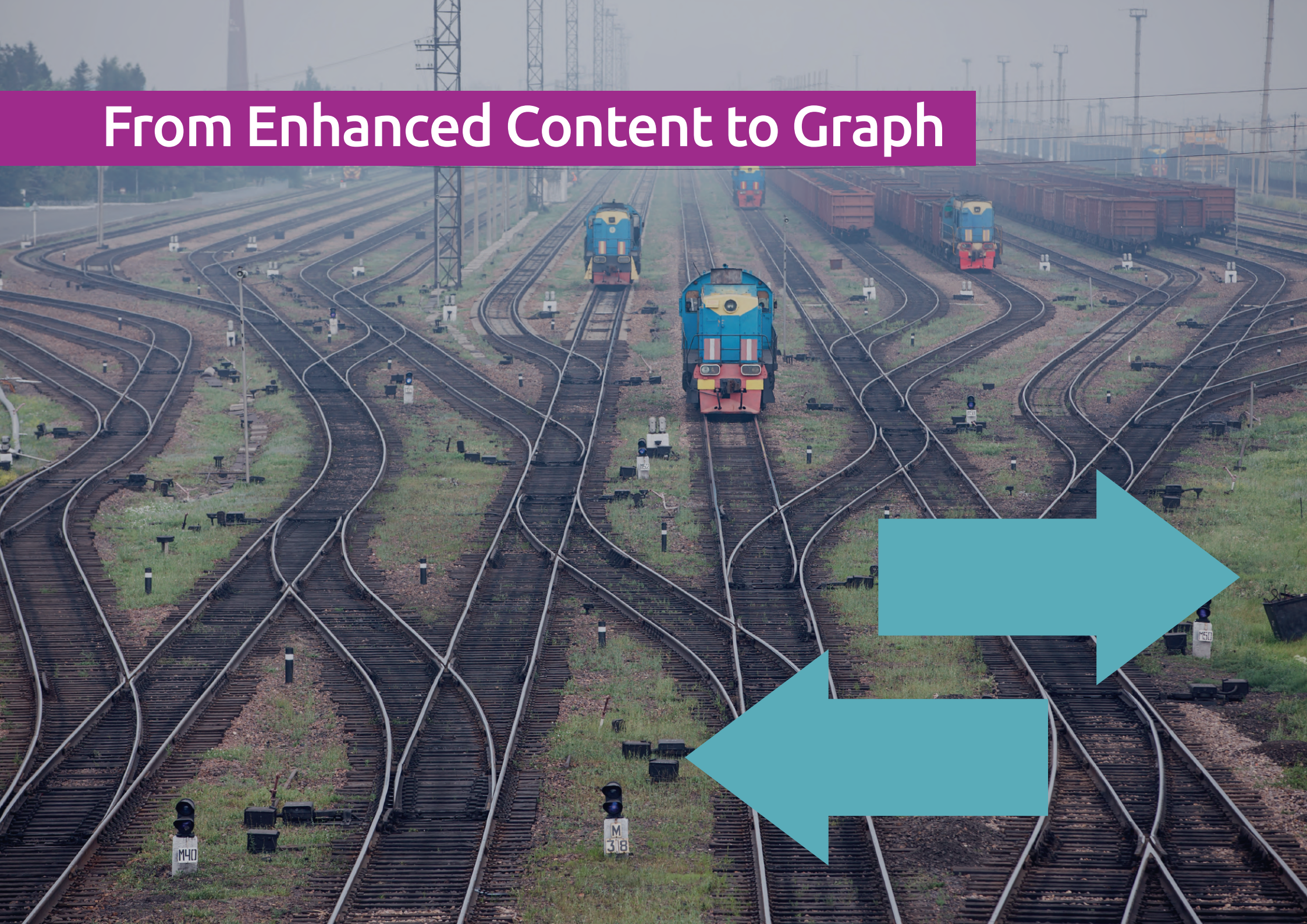
On the topic of deep learning

We should say that it is far from a magic bullet. It performs well on nicely bounded tasks, when enough data is available for training (sometimes you need millions of examples), but it's not suited to large domains such as Question Answering. Hence, letting the bot learn by itself does not really apply in the situation where the knowledge is broad and the users few. The scarcity doesn't allow the pure machine learning approach to work.

For more on the topic, the article "Deep Learning: A Critical Appraisal"[1] from Gary Marcus outlines well the limits of deep learning, which explains why the scientific community is now looking at new approaches for AI.

1. https://arxiv.org/abs/1801.00631

From Enhanced Content to Graph



Content fragmentation

Content enrichment is a necessary step for making textual content, originally intended for humans, consumable by algorithms, bots and machines. No matter how complex it seems to be, it's totally within reach as it uses three practices that are already common among content creators. We call the use of these techniques the Content Value Path:

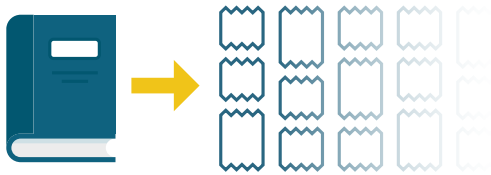
1. Structural fragmentation
2. Surface semantic enrichment
3. Deep semantic enrichment

We will review these three steps below and show how they tie in closely with one another, and how their combined effect opens up new opportunities and gets us closer to question answering.

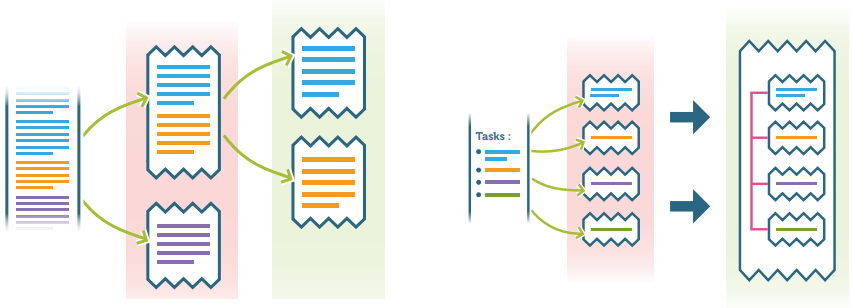
For more information on the term "semantic", which is used a little differently here than in common speech, see the last chapter: *What Does "Semantic" Mean?*

Structural Fragmentation

Structure is meant here as part of the structured content authoring concept, a widespread practice in technical documentation that consists of breaking the content down into little pieces (from a few lines to a few paragraphs) called components or topics. These are then assembled via maps (similar to tables of contents; a map is a list of pointers to topics) to create the final content. The corresponding standards and tools are well known, such as Docbook, DITA or S1000D. This approach runs contrary to writing long, unstructured documents using word processing tools. It was designed to optimize the production and maintenance of large bodies of documentation by writing in parallel, avoiding duplicate content by recycling components, facilitating modifications, reducing translation costs, etc.



Note that, in this productivist approach, the granularity of the components is determined by production issues and is not necessarily correlated with the content itself, i.e., the subjects broached in the text. In our **Content Value Path** approach, the **breakdown of the topics must be aligned with the subject** because we **need consistent and complete grains of information**. Excessively long topics that deal with several subjects must thus be broken down to a more granular level. Conversely, excessively small components (such as a phrase or fragment) resulting from a given authoring technique must be assembled within intermediate maps that are not necessarily intended for publication, but that serve to define this consistent level of information.



Large topics dealing with multiple subjects should be split further.
Very small topics must be grouped with intermediate maps.

Content enrichment

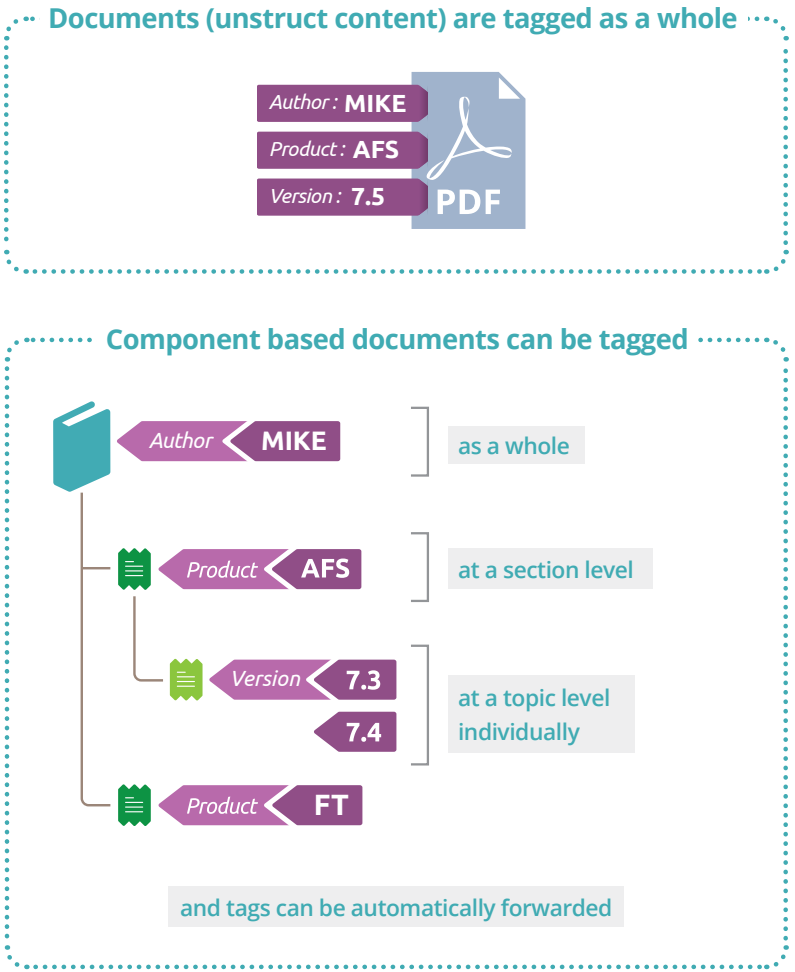
Why is breaking down our text necessary to help computers read? Because it enables the respective technologies and algorithms to work with complete grains of information, and thus to target the elements needed to answer a question more effectively and unambiguously. But to do that, we must still add the metadata and semantics layers, both on the surface of documents and deeply within the text.

Surface Semantic Enrichment

Metadata is data that describes other data. For example, the date a component is created, author, and publishing status, are management metadata. Classical metadata in the context of technical documentation are product, version, type of task, targeted audience, etc.

Metadata can be dates, numbers, free keywords or, more typically in documentation, labels taken from controlled lists: flat or hierarchical lists (taxonomy). This metadata can be positioned on the topics or maps (in which case it is conveyed to all the topics included in the map).

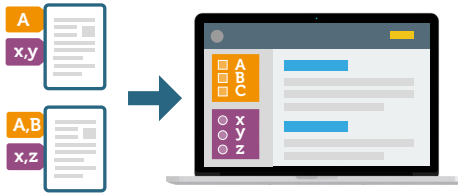
The link between structure and metadata is clear: for metadata to be meaningful and usable, it must apply to the entire topic unambiguously. Thus, for example, if a topic contains information on both the installation and maintenance of a product, it will have to be split into two distinct topics, one containing information specific to the installation, the other information specific to maintenance so that each can be labeled more accurately.



Content enrichment

The practical issue concerns the choice of metadata: which metadata, with which values? This will depend on your products, your content and how you want the content to be used. Here are some typical use cases:

- In a search engine, create filters – also called facets – so that users can refine their search, for example by only retaining responses specific to their product.
- When reading content, dynamically filter a manual by removing sections that don't apply. Example: in an installation guide, display only generic instructions and these corresponding to the specific version of the product; remove the elements corresponding to other versions.
- In online Help, display contextual information linked to the exact version of the product and its configuration.
- For a maintenance operation, display a bar code and an error number on the machine. The operator will respectively scan and enter them to access the maintenance procedure. The bar code is translated into a list of the machine's constituent subsystems and applicability conditions, all of which represent the metadata used to filter the content.

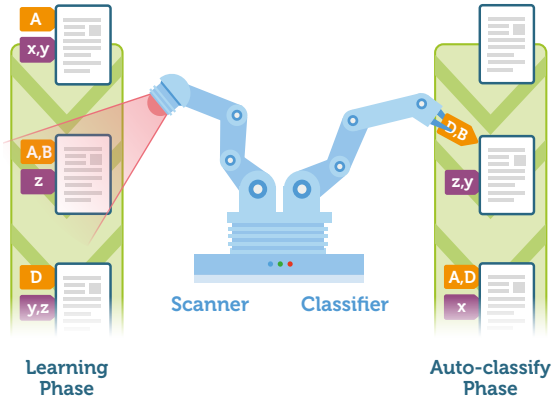


Metadata used as filters in the search experience

If you are starting from scratch with your metadata, or if you have already gathered some metadata and are wondering how to proceed, follow these steps:

1. First, define some use cases via scenarios that involve typical users based on standard techniques involving characters and storytelling.
2. Next, identify the metadata needed to support these scenarios: which criteria are necessary to extract the content with filters?
3. Match the metadata with the content. Here, you may have to adapt your content's granularity, as mentioned in the Structure step.

This last step may appear daunting if you have thousands or even tens of thousands of topics. This is where technology steps in. Today's automatic classification algorithms, which use the latest technological advances in artificial intelligence, are extremely accurate. They are able to learn from a representative set of manually labeled topics (supervised learning phase), and then proceed on their own (automatic classification phase). Consequently, with just a few hundred pre-labeled topics, you can tag thousands or even millions of topics in a matter of minutes. You can do the same for content from any other source (wikis, knowledge bases), and thus benefit from a fully aligned body of documentation. Here too, we must stress the need for topics with the right level of granularity. The more focused the topic's content is (in particular concerning topics used for learning), the more precise will be the automatic labeling.



The automatic classifier learns by scanning a set of already labeled content and then tags any content with accuracy.

Content enrichment

Deep Semantic Enrichment

We have given our machine smaller, bite-sized pieces of text (topics), we have labeled them with tags indicating what they are about (product, version, and so on). The last step in the Content Value Path is that of deep semantic enrichment.

When we learn to read, we have to discover what sentences are, paragraphs, lists, but also to recognize dates, people or product names, and so on. To teach our machines to read, we need to label these textual entities, and program the computers what to do with each piece of text.

The first aspect of deep semantic enrichment has to do with informative tags to express the structure of the content. As an example, consider the list of steps involved in a maintenance task. It would be written as an ordered list in an HTML wiki or using a word processing tool:

```
<p>Change cartridge</p>
<ol>
  <li>Step 1: Do this</li>
  <li>Step 2: Do that</li>
  <li>Step 3: Finish</li>
</ol>
```

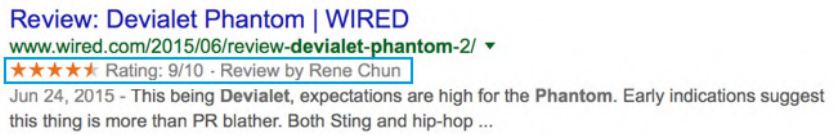
This would probably be rendered as a numbered list, which would be helpful to an educated human reader. A machine reader, however, would need more. If you are going to program the machine to perform a task, you need to tell it explicitly that these are “steps” in a “task”. In DITA XML, the same content would be written like this:

```
<task><title>Change cartridge</title>
<steps>
  <step>Do this</step>
  <step>Do that</step>
  <step>Finish</step>
</steps>
</task>
```

Even if this level of semantics represents a significant contribution, it is insufficient in our Content Value Path, as it is limited to the structural aspects of documentation. It must therefore be improved by a form of semantic enablement that would identify the informational elements specific to the business context, such as products, references, components, technologies, measurements, and dates.

This technique is widely used for Web content in the context of Search Engine Optimization (SEO). It involves labeling the text using tags that allow algorithms to unambiguously identify and extract information such as the name of persons or products, dates, quantities, references, events, the software application or version, the type of task described (such as installation or maintenance), or the required level of expertise. This is possible because the web pages have specifically marked this information via tags (see schema.org, RDFa, JSON-LD and Microdata).

In the example below, we can see that, in addition to text, Google displays structured data: rating and reviewer.



Content enrichment

The same principle can be applied to technical documentation. For example, behind the sentence

> To add module X23b from Acme you need to increase the output power of the converter to 20W by using the calibration tool PV37.

one can see the relevant entities:

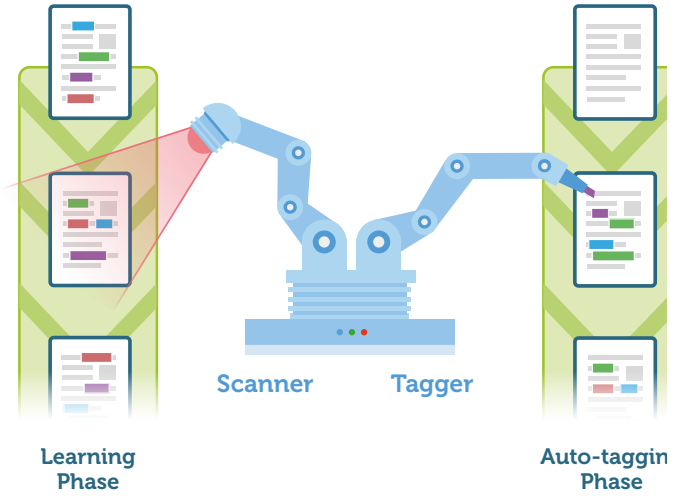
> To add module X23b from Acme you need to increase the output power of the converter to 20W by using the calibration tool PV37.

And the actual text would include markers that look like this:

> To add module **<partref>** X23b **</partref>** from **<brand>** Acme **</brand>** you need to increase the output power of the converter to **<measure unit="W">** 20W **</measure>** by using the calibration tool **<toolref>** PV37 **</toolref>**.

Then the question is: how do you go about labeling all your content? Will you have to insert all these markers manually? This would appear to be a superhuman and unrealistic task, especially if you have thousands of pages of documentation.

Here again, technology comes to the rescue. Thanks once more to the progress made in machine learning, algorithms can automatically perform this marking with a very high level of precision. They only need a few dozen manually tagged topics to learn before being able to pursue this on their own. This enhancement task is usually performed outside the content management system (CMS) and is integrated into the publishing system, which performs it on the fly during the analysis of received content.



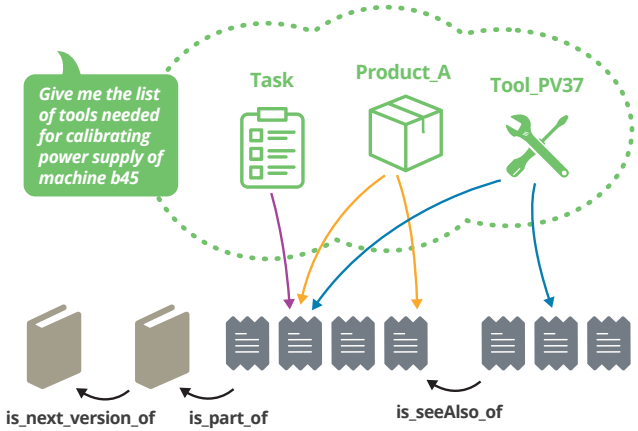
The automatic tagger learns by scanning a set of already marked content and then highlights the key concepts in any text with accuracy.

Structure, topic-level (surface) enrichment and text-level (deep) enrichment of content – these are the three steps inherent to the Content Value Path.

Connecting to rich content

We have described the strategy to chunk the documents into small fragments and to enrich the text by adding onto it, and into it, tags that express the implicit semantics. After this phase, the text contains indicators of specific entities of interest: products, components, features, tools, tasks, etc.

The next step is to generate a graph that charts the relationships between the **document objects** (maps, components) and the **semantic objects** that are derived from the enrichment phase. This Graph is useful for giving formal meaning to the relationships. By 'formal', we mean it supplies a vocabulary and grammar that depicts how the different objects relate to each other. For example: topic X contains a procedure on how to fix product A, and one step of this task mentions tool PV37. By making these relationships explicit, we see how it becomes possible to respond to complex questions such as, "What is the list of maintenance procedures involving component X23b", or, "Give me the list of tools needed for an intervention on machine b45."



The Smart Graph represents the relationship between the documents and the entities.

The approach used for modeling the graph is not important, and lies beyond the scope of this document; even extracting and storing the graph is not mandatory. What matters the most is the technology we use for indexing the content, the semantic objects, and the relationships all at once.

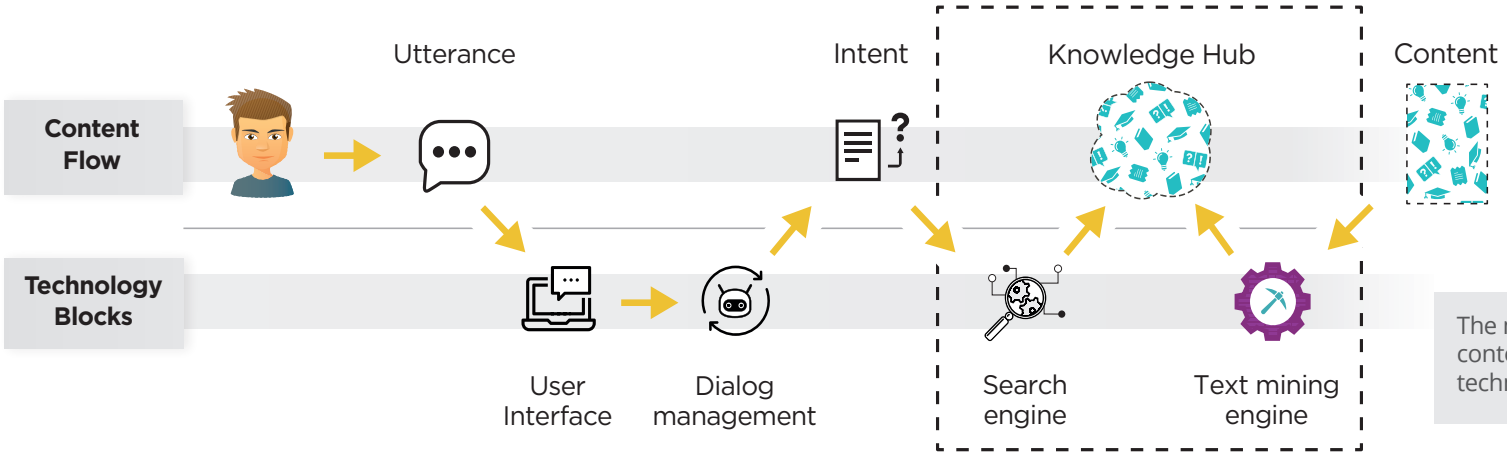
Indeed, we need to **index** this fine-grained enriched text with a search technology that can index not only words, but also the outlined concepts and typed entities. This requires a new type of search technology, beyond the classical full-text search[1]. We need a search technology that allows the dynamic use of any type of parameter for filtering and ranking results that are not documents, but knowledge objects. By dynamic, we mean it has the capability to build a search equation that contains any attribute of the objects we are considering, and that the equation would determine how to sort the results. It is important to note that when dealing with chatbots, we are looking for "one" result only. As opposed to regular search on a web site where we try to have a decent first page of results (giving the user the burden of figuring out which result is "best"), which is already a challenge, when dealing with chatbots, only the first result of the list is considered and used by the bot, so we must ensure that the first result is the good one. This is why all the existing full-text oriented search engines can't cope with the mission and are far from being the solution.

1. Typical examples of these limited search engines are Lucene based SolR or Elastic Search

Connecting to rich content

Connecting chatbots to rich content requires a more flexible technology that allows you to transform a complex intent into a search equation without any limitation, which means without the constraint of a predefined indexing and ranking model.

This innovative breed of search technology is very new and still rare, but it exists—we have it within Fluid Topics—and it opens totally new possibilities for handling user queries. Its probabilistic nature permits the transformation of user intents as provided by the upper framework into formal queries, through the application of a meta-model. The results themselves are more relevant, containing not only focused text fragments, but also knowledge objects, and it becomes possible to reply to questions such as "Give me the list of tools needed for calibrating the power supply of machine b45".



The missing piece is a mix of enriched content and bespoke new search technology.

So now what? How to set up your first chatbot

Since you've reached this point in the document, you see that setting up a chatbot to deliver product and technical support is easier than it has ever been due to new technology and the profusion (if not the confusing overabundance) of frameworks. But there is still a high risk of failure because of the very nature of technical content. This is why Gartner analysts say[1] that virtual agents are reaching a peak of inflated expectations but should soon start to slide into the trough of disillusionment.

This doesn't mean you should wait, but rather start tactically. To plan your approach, use the 80-20 rule: 80% of your users' questions should be covered by 20% of your content. The exact percentages may vary, based on your industry, products, and context, but the principle stands. Focusing on that 20% of your content is obviously a good idea, but how to identify it is the trick. This can be done in a few ways; here are three angles that can be fruitful.

Analytics: if you have precise metrics of what your users read, and, if possible, what they were looking for when they chose that content (in other words, what keywords did they use in their search), that will give you a reliable idea of what content is used the most. The keywords represent their intent, and the content they read could be fragmented, tagged, and made more conversational; this will make it transferable to chatbot dialog. As we describe in our white paper, "Analytics – moving from cost to value," generating precise and unbiased metrics is not that simple, and using flawed numbers could lead to excessive or even mistaken work.

Support tickets and answers: if you already use a helpdesk tool, this can be a valuable source of information. It may not contain the most trivial questions, because users simply go to your documentation for those (which leads back to your analytics; see the previous point). But if you examine the problems and questions described in the trouble tickets you will be able to discern which parts of your content would be most beneficial to focus on.

Already existing KB: usually, support agents have already done part of the work of studying the tickets, and, to speed up the reply process, they may have created a small knowledge base (Questions & Answers) covering the most frequent subjects.

Once you have identified what subjects and questions come up the most, you know what content should be adapted to feed your chatbot project.

Concrete next steps are:

- Manually design a few dialogs (sets of questions and answers) in a conversational way, feed that into the chatbot framework and run an experiment with real users.
- Prepare a larger set of questions for training the intent extraction module of the chatbot.
- Create a thesaurus of synonyms corresponding to the terms appearing in the questions.
- Identify the 5% to 20% of your content that corresponds to the most asked questions. Optimize that content (adapt the grain size and tag it).
- Index that content with a semantic search technology able to leverage the structure, the tags, and the thesaurus. Connect the chatbot to this search engine endpoint.
- Scale to your entire body of knowledge with technology: automatic content fragmentation and machine learning based tagging will do most of the work for you.

Now you have an approach for taking a user's question and responding. If you've set up your body of knowledge and search engine well, that response will be accepted as an answer to their question.

1. https://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/

Deep Dive: What does “semantic“ mean?

We have been talking a lot about *semanticizing content*, and maybe you wonder what it means and what it really entails, particularly in the context of technical documentation.

Before illustrating this with a concrete example, let’s look at a dictionary definition of “semantics”: *arising from the different meanings of words or other symbols*.

If you are a human being, you are reading this text in the traditional way, by looking at the words and leveraging your experience and education to build a model of meaning in your consciousness based on this text. Now, if we want a machine to work with text or images, it becomes extraordinarily difficult, unless, as explained by Tim Berners-Lee (the inventor of the World Wide Web), the **information is given well-defined meaning**.

Information is given meaning? This sounds strange… After all, doesn’t information already have meaning? Isn’t that the point of information? Yes, it has meaning, to me, because as a human I have experience and an education, which allow me to build internal abstract representations, so that I can extract meaning from text by mapping what I read to these models.

A good illustration of how we go from text — or images — to meaning is Magritte’s famous painting bearing the legend, “This is not a pipe” (the actual title of the painting is “The Betrayal of Images”). If you show this image to anyone, they will say, of course it is a pipe, instantaneously associating the image with their internal representation of a pipe and automatically connecting to various other related ideas (smell, smoke, tobacco, grandfather …). But the artist reminds us that we’re going too fast, because it’s just a painting of a pipe, not an actual pipe. But that’s how it works in our brain: we map the image to internal representations and memories.

Magritte, René. *La Trahison des Images.* 1929. Oil on canvas. Los Angeles County Museum of Art.

A sad little computer, however, has no such experience, and needs help. The information needs to be labeled, and the computer must be given instructions on what to do with those labels. The word “pipe” has no meaning for a computer. It just sees four letters “p-i-p-e” (even worse, it sees four bytes in ASCII, #70697065). It is far from meaning and action.

Now let’s see the implication in technical documentation. Take an example of a fragment, explaining how to calibrate a thermostat (this example is heavily edited; see the original here, at RJ Gas Appliances[1]).

First, let’s present it in a way nobody would ever use – a simple string of text.

1. http://rjgasheating.co.uk/calibrate-thermostat/

Deep Dive: What does “semantic“ mean?

All the information is there. If you just had this, you could probably calibrate your thermostat. You understand that the first two sentences are title and subtitle introducing the rest – it does not give you an instruction, but rather it characterizes what follows. You realize that each sentence is a step, to be completed before going on to the next sentence.

The way it is presented on the web site is a lot more helpful, with the simple addition of some formatting:

How to Calibrate a Thermostat

Learn how to calibrate a thermostat following these easy steps.

- Find out what type of thermostat you have: one contains magnetic contacts and a calibration screw, and the other employs a mercury switch.
- Clear the area around the thermostat. Check for things that might be interfering with your thermostat’s reading.
- Check the thermostat temperature. Turn the thermostat on and place a thermometer on the wall a few inches away for 15 to 20 minutes. Compare the thermometer’s reading to that of the thermostat. If there is a difference of more than one degree, perform the recalibration.
- Calibrate: Screw thermostat: Turn the screw left and right until the contacts open; turn it again until the contacts close.
- Calibrate: Mercury-switch thermostat: Check that the thermostat is completely level.

So, we made the title obvious, and indicated with formatting that there are multiple instructions. Each bullet point shows a step of the procedure, and this is something that you recognize from past experience, having read many lists of bulleted instructions. So, you are already unconsciously semanticizing the text by turning each bullet into a step. Further, still based on your past experience, you assume that steps have to be executed top to bottom. You are giving it meaning. Still, you need to read and understand the last two steps to realize that you only do one of them, depending on the result of the first step.

So let’s add some helpful semantics... helpful to people, to reveal the implicit expertise of the reader, and design this to-do-list in a less ambiguous way.

How to Calibrate a Thermostat

1. Clear the area around the thermostat. Check for things that might be interfering with your thermostat’s reading.
2. Turn the thermostat on.
3. Place a thermometer on the wall a few inches away for 15 to 20 minutes.
4. Check the thermostat temperature. Compare the thermometer’s reading to that of the thermostat. If there is a difference of more than one degree, perform the recalibration (see 5 and 6).
5. Find out what type of thermostat you have: one contains magnetic contacts and a calibration screw (see 6a), and the other employs a mercury switch (see 6b).
6. Calibrate
   a. Screw thermostat: Turn the screw left and right until the contacts open; turn it again until the contacts close.
   b. Mercury-switch thermostat: Check that the thermostat is completely level.

The numbering makes it clear that you should start at the top and work down, and not, say, do the steps at random, or pick just one or two. By adding the (a) and (b) in step 6, we try to insert more semantics, hoping that the reader would understand that when they get to step 6, they pick one or the other, depending on what kind of thermostat they have.

We have made things more explicit, hence relying less on the reader’s experience, but this still isn’t helpful to a software program, such as a virtual assistant that could guide you if you would ask it how to do this calibration (think Amazon Alexa or Google Home).

Deep Dive: What does “semantic“ mean?

For that to happen, we need to be painfully explicit. We must tag the text in a way that indicates to the computer that this content refers to a task, and that we want it to exhibit the steps; formatting will not do the trick, we need clear, unambiguous labeling. Using structured authoring (such as DITA or other XML based format), it could look like this:

```
<procedure>
<title>How to Calibrate a Thermostat</title>
<step>Clear the area around the thermostat. Check for things that might be interfering with your thermostat's reading.</step>
<step>Turn the thermostat on.</step>
<step>Place a thermometer on the wall a few inches away for 15 to 20 minutes.</step>
<step>Check the thermostat temperature. Compare the thermometer's reading to that of the thermostat. If there is a difference of more than one degree, perform the recalibration.</step>
<step>Find out what type of thermostat you have: one contains magnetic contacts and a calibration screw, and the other employs a mercury switch.</step>
<step>
<choices><intro>Calibrate</intro>
<choice device-type="screw">Screw thermostat: Turn the screw left and right until the contacts open; turn it again until the contacts close.</choice >
<choice device-type="mercury">Mercury-switch thermostat: Check that the thermostat is completely level.</choice >
</choices>
</step></procedure>
```

It is now clear that this is a procedure (it's right there in the tag), that the steps are, in fact, steps, and that you can only select one of the final choice instructions. This is semantically enhanced. Now, a computer has a fighting chance to know what this is about. It could display it in a helpful way.

Of course, we are anthropomorphizing here. The computer can't “know” anything. But it could be programmed to respond to the “step” tag by displaying the step in a particular way. You could choose numbers or bullets; you could tell it to display them one by one if your screen is small; you could tell it to read them to you if your device is a speaker. If it knows the model number of your thermostat, it could (and really should) only display the relevant instruction (this is what Dynamic Delivery is about); if I have a mercury switch thermostat, why do I need to see the screw thermostat instruction?

We could go one step further and add more semantics to the content:

```
<procedure>
<title>How to Calibrate a <device type="tst">Thermostat</device></title>
<step>Clear the area around the <device type="tst">thermostat</device>. Check for things that might be interfering with your thermostat's reading.</step>
<step>Turn the <device type="tst">thermostat</device> on.</step>
<step>Place a <device type="tmr">thermometer</device> on the wall a few inches away for <duration sec="900">15 to 20 minutes</duration>.</step>
<step>Check the thermostat temperature. Compare the thermometer's reading to that of the thermostat. If there is a difference of more than <measure unit="degree" val="1">one degree</measure>, perform the <action_step id="calib">recalibration</action_step>.</step>
<step>Find out what type of thermostat you have: one contains <device type="tst/screw">magnetic contacts and a calibration screw</device>, and the other employs a <device type="tst/mercury">mercury switch</device>.</step>
<step id="calib">
<choices><intro>Calibrate</intro>
<choice device-type="tst/screw">Screw thermostat: Turn the screw left and right until the contacts open; turn it again until the contacts close.</choice >
<choice device-type="tst/mercury">Mercury-switch thermostat: Check that the thermostat is completely level.</choice >
</choices>
</step></procedure>
```

Deep Dive: What does “semantic“ mean?

It becomes less and less easy for a human to read it in this form (even though the formatted text output would still look the same), but now see how a computer could leverage this content further and become able to answer questions such has “*Tell me how to calibrate a mercury switch thermostat*”. The computer may seem intelligent, but you can see that everything it needs to find the right content is here, in the content, not ambiguous even for a dumb computer with no knowledge.

Finally, we could write the same procedure in a programmatic way:

```
function Calibrate_TST(device d) {
        Turn_on(d);
        Wait(900);
        t = Measure_Temperature(from_probe);
        if (abs(d.temp – t) < 1) return(no_action_required);
        if (d.type == “tst/screw”) return Calibrate_TST_Screw(d);
        else if (d.type == “tst/mercury”) return Calibrate_TST_Mercury(d);
        else return(unknown_device_type);
}
```

This program could be used by a robot to do the calibration task: it is totally explicit. It conveys not only semantics but also the actions corresponding to each step. But it is in a form that is nearly impossible for humans to read, and it is even difficult to use it to guide a human or generate a readable text output.

To Wrap Up

Semantics, at least in the context of computer text handling, concerns the role the text plays, married to the actual content of the text. With tags and metadata, we can name that role and state it explicitly, whether it is title or step or choice or anything else. When you combine the content of the text with a clear statement of the role the text plays, we get closer to explicit “meaning.” This permits us to train machines to work with the text in a way that is coming closer and closer to something we might call “understanding.”

ABOUT THIS GUIDE

The challenge of new channels for tech content delivery

This white paper explores the complexity of feeding new delivery channels with textual content, whether XML, legacy, Wikis or anything else. It particularly focuses on the challenge of chatbots and reviews the various approaches that the maturity of technology allows.

Table of contents

What's behind Fluid Topics

Fluid Topics is a perfect name for a dynamic publishing product. "Fluid" conjures images of effortless motion, smooth and easy flow, and graceful simplicity in movement or execution. "Topics" evokes conversation, communication, learning, thinking, reasoning, education, and ideas.

Together, **Fluid Topics** conveys a smooth action or movement of reaching for and shaping information, quickly and effortlessly without limit or barriers.

That is what we do.

Contact us:

For technical questions:

fluid.support@fluidtopics.com

Website:

www.fluidtopics.com

Blog:

www.fluidtopics.com/blog

Twitter:

@fluidtopics

Tech Content Delivery

The Challenge of New Channels

www.fluidtopics.com
info@fluidtopics.com

Revision: November 2020

Fluid Topics