CONTENTS

OVERVIEW

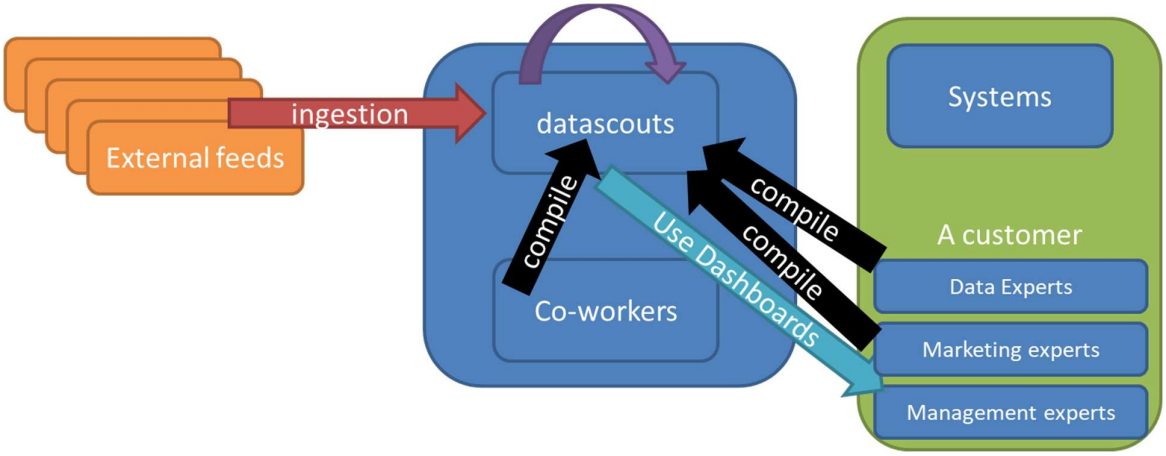
ACTOR PROFILING

The DataScouts platform is a stable environment offering Ecosystem Intelligence, combining actor profiling with a series of interactive dashboards. Data ingestion, unification and aggregation is the core capability of the current platform.

At this moment, 23 pilots are active on DataScouts. We have 9 paying customers and 1 partnership contract is in place with Startups.be to provide startup enablers and accelerators in Belgium access to their data via the DataScouts platform.

Roche (ES), Ford (DE) and ING (BE) are using DataScouts for sharing knowledge and capturing feedback from colleagues about the innovators that are relevant to their corporate venturing strategy or the innovation space they are active in. We are running pilots with Teamleader, ClearChannel and Lansweeper to validate the fit for competitive profiling.

Our target customers are large (service oriented & digitally enabled) SMEs and corporations who are looking for competitive intelligence in developing a business plan for sustainable business growth as well as federations and associations representing industry aiming to monitor their market and the members they represent.



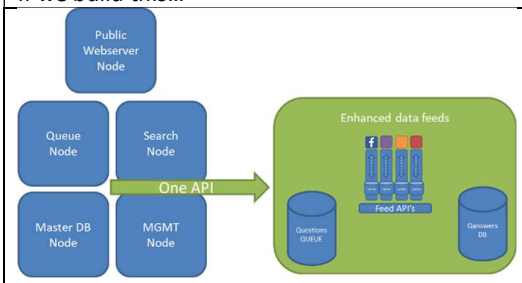
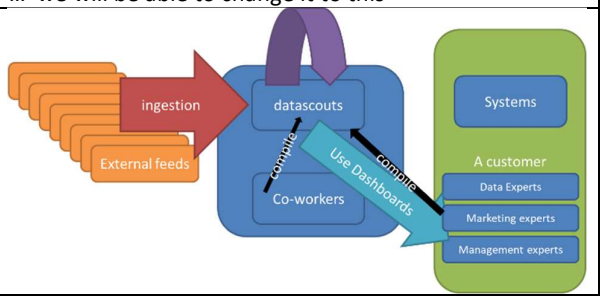
We like to call this concept “Actor profiling”

OUR CHALLENGES

ENHANCING OUR DATAFEED CAPABILITIES > MACHINE ASISTED COMPETETIVE INTELLIGENCE

Generating an initial data set for mapping the competitive landscape is partly a manual process, it takes quite some effort and time to generate a less than perfect competitor profile. As a result, DataScouts is not scalable today, our growth is hampered by the manual effort needed to onboard new customers.

WU1 will allow us to make a leap forward in terms of automating the data collection and competitor profiling process. We will upgrade the current IT infrastructure to be performant, secure, scalable and enhance the product architecture to provide a strong generic engine to generate an ecosystem specific data feed combined with smart classification capabilities. We will incorporate Named Entity Recognition and Active Learning in order to capture as much as possible intelligence within an ecosystem with its specific boundaries in terms of actors, geographies, markets, technologies.

If we build this…	… we will be able to change it to this
	

As you can see none of the major processes currently supported will be changed, but their weight will have.

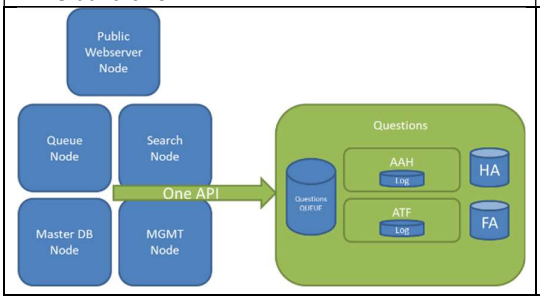
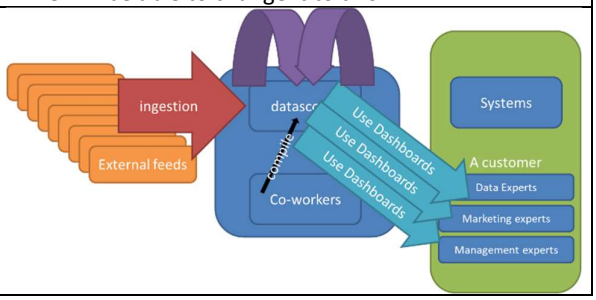
- We want to increase the automatic unification and aggregation processes so that they can alleviate all the human compiling and data crunching work involved in
    - Setting up a landscape (automatic suggestion of new actors)
    - keeping it up to date (automatic enrichment)
    - getting the data accurate (better enrichment)
    - adding more relevant data (more enrichment)
    - removing non-relevant data. (suggesting removal of non-relevant actors)
- Increase the number of feeds (by standardising the technical setup)
- More granularity on configuration per ecosystem (by utilising the extra standardisation of feeds)
- Calculate more intelligence metrics to create more and more valuable reports

This would bring us to state where we can call it
“Machine assisted competitive intelligence”

ENGAGING HUMANS TO ASSIST AUTOMATION > EXPERT ASSISTED COMPETETIVE INTELLIGENCE

Competitive analysis is a continuous process. In addition to the methodology for collecting data about competitors, market knowledge is required to ensure expert curation of the collected information. We are convinced that competitive analysis is a core competence of successful businesses. The wide adoption of DataScouts in an organization is therefore a must-have to gather and curate data in order to accurately reflect the competitive landscape and serve as input forstrategic decisions.

Although we dream of 100% accurate data feeds without any human intervention, we need to be realistic. Artificial Intelligence (AI) and its sister disciplines of machine learning, cognitive computing, sentiment analysis and neural networking have a problem: they're artificially created through the power of software developers' algorithms. They do not own the capacity to make complex decisions. We strive towards 80% machine generated data, 19% human enrichment and curation and 1% randomness as perfection is impossible to reach. We will therefore build and test a series of channels to gather the feedback of humans, experts in the fields in order to answer questions and solve ambiguities. We assume to be able to improve the quality & accuracy of the competitor profiles by putting a human in the loop. WU2 focuses on the interaction of humans, as much as possible personalised, contextualised and distributed over different employees and touch points, in order to keep the burden to enter data as low as possible. We will experiment with gamification components as team challenges and badges to validate the impact of a fun factor to data crunching

If we build this…	… we will be able to change it to this
	

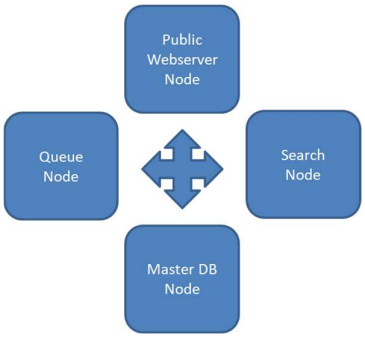
All major processes still exist, but can now be handled very differently

- Original intake can be mostly automated by asking the right questions to the right people.
  A simple initial pre-sales intake can be enough to get it started since at this point we only need to gather information about certain entities and the geographical scope they want to keep an eye on and we fill in a handful “known competitors” .
  We then make sure that the reference actor (which is the customer itself) and the handful of examples are correctly linked to the correct entities and we can let the application look for more competitors itself.
- The application will be able to suggest actors to add and remove pro-actively during the entire lifecycle
- The application will automatically do all it can to enrich everything it is allowed to enrich instead of having internal colleagues having to think about how best to enrich this specific dataset
- By including the human feedback in the loop we can go even further in determining value of certain feeds , and certain types of human input.
- The amount of “insight metrics” can be greatly expanded on once again which will allow us to create custom tailored dashboards for each of the identified user groups within our customers
  - Data Experts
  - Marketing Experts
  - Management experts

And this we would like to call “Expert assisted competitive intelligence”

HOW ARE WE GOING TO DO THIS

From a high level perspective we currently have the following technical setup in place which already supports our current customers.



All these “nodes” are a collection of a specific technical domain which is accessible by any of the other “nodes” by web based API’s.

1. Public Webserver node : The webapplication itself , fully developed internally
2. Queue node : A “beanstalkd” implementation which we can use to defer workloads to a later point in time.
   Here is how “Beanstalkd” describe it themselves and I couldn’t find a better description myself.

   “

   Beanstalkd is a big to-do list for your distributed application. If there is a unit of work that you want to defer to later (say, sending an email, pushing some data to a slow external service, pulling data from a slow external service, generating high-quality image thumbnails) you put a description of that work, a “job”, into beanstalkd. Some processes (such as web request handlers), “producers”, put jobs into the queue. Other processes, “workers”, take jobs out of the queue and run them.

   ”

3. Search node : an implementation of “Elasticsearch” , which again “Elasticsearch” describes perfectly what it’s for.

   “

   ## Fast, Incisive Search against Large Volumes of Data

   Conventional SQL database managements systems aren't really designed for full-text searches, and they certainly don't perform well against loosely structured raw data that resides outside the database. On the same hardware, queries that would take more than 10 seconds using SQL will return results in under 10 milliseconds in Elasticsearch.
   A user expresses an ES query with a simple language, **Query DSL**. A query examines one or many target values, and scores each of the elements in the results according to how close they match the focus of the query. The query operators enable you to optimize simple or complex queries that often return results from large datasets in just a few milliseconds. The Elasticsearch design is much simpler and much leaner than a database constrained by schemas, tables, fields, rows, and columns.
   “

   Logically we use this database node to store the types of data that are more efficient to store and search in this type of database technology

4. Master DB Node : an implementation of “MariaDB”. Which is a classical “Sequential Database “ product comparable to “MySQL” or “Microsoft SQL server” etc…

   In which we store all data that is efficient to store and retrieve in this type of database technology.

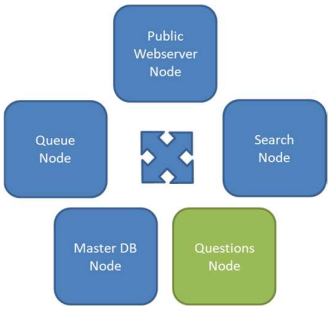
Modules

-

This base design has been the platform that was needed to fullfill all functional requirements until now, but based on the experiences we have now had with

1. evaluating different types of ecosystems
2. analysing all business processes involved for ourselves and our customers
3. evaluating the market readiness to outsource some of these processes

We have now come to the conclusion that we need to add 1 more Node to enable the advanced functionalities we are now looking for , and we have called it “The questions node”.



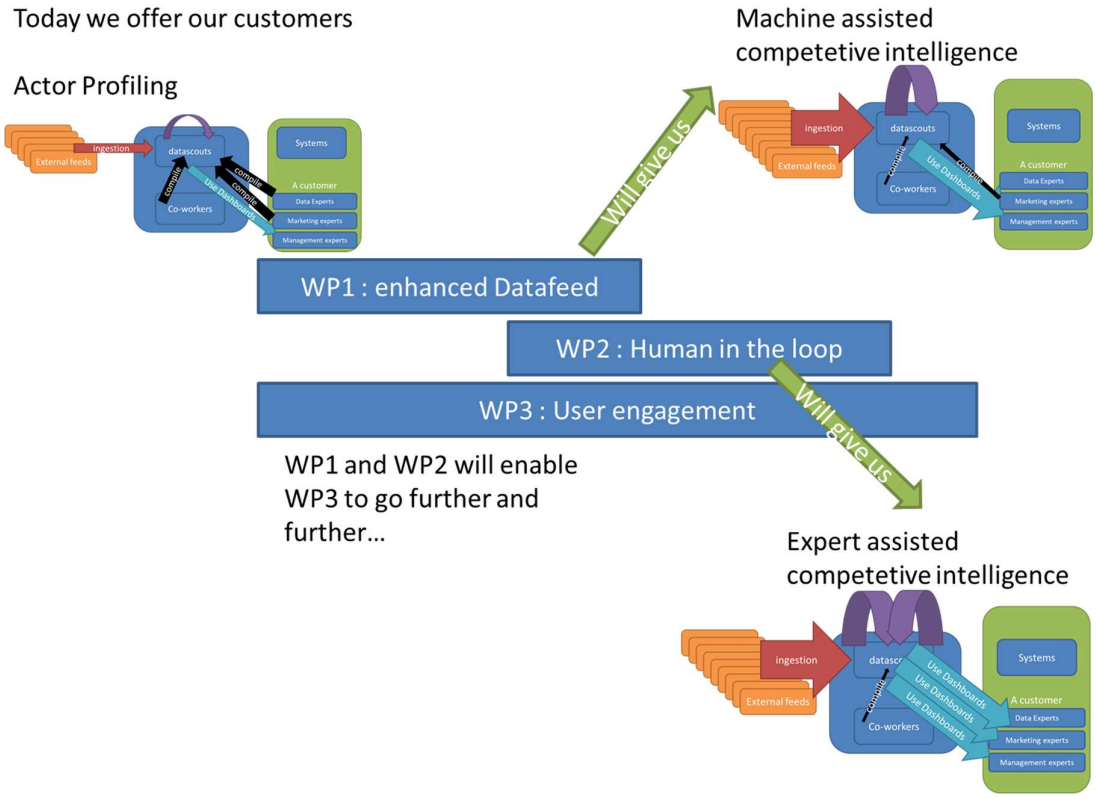
In essence this “Questions node” is a functional island with two subcomponents that both handle “Questions”.

- “Ask a human”
  We have noticed that we need to have a well automated way of checking certain values with humans and logging the results of these questions and how they were answered by whom.
- “Ask the Feeds”
  Although we already have numerous feeds that our filling our various databases, we have noticed that by redesigning our current feeds concept into one where we handle each datapoint as a separate “Question” we can engage numerous AI technologies in a standard way.

Whenever the application encounters a point in its process where it “requires input” from either a human person, or an external datafeed , then the application can “Ask” this “question” to the web API’s of the Questions node which will then do its best to answer that question within the allotted timeframe (which will depend on the type of question)

The base design of the questions node allows us to continuously cross-reference all “answers” made by automation with those of humans which will enable us to reach a scalable functional concept of “Active Learning” for the entire application.

HIGH LEVEL TIMELINE



WP1 : ENHANCED DATA FEED

FUNCTIONALITY

The first functionality we want to have the “questions” node perform is to be able to answer questions about individual “datapoints”.

A typical simple question might be

“What is the address of this VAT number according to KBO?”

But we want it to be able to handle more complex questions as well like

“What extra information is there available about the company with this VAT number”.

The “complex” questions can always be subdivided into several “simpler” questions in a specific order so this question could be subdivided into several smaller questions in a specific order

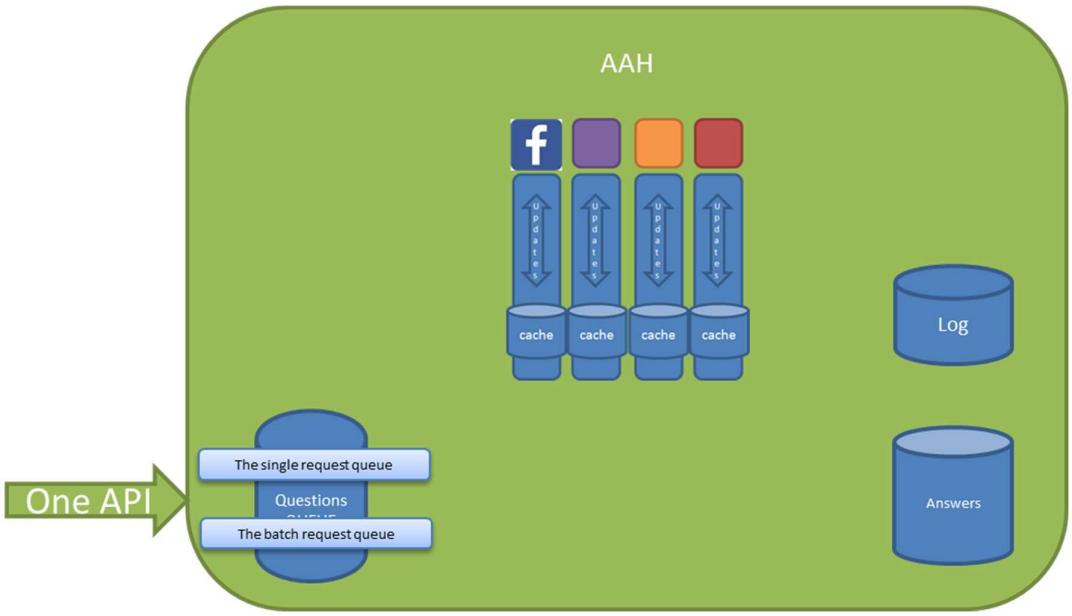
“What extra information is there available about the company with this VAT number”.

Question	**Answer**

Which feeds allow searching on VAT number?	KBO & OpenCorporates & etc…
Which datapoints can KBO give me based on this VAT number search query?	Name, Address , type of company , sector , etc…
Which datapoints van Opencorporates give me based on this VAT number search query?	None (apparently Opencorporates doesn't know this specific VAT number)
According to KBO , what is the address, name, type of company, etc… of this VAT number?	- Name : "A company" - Address : "address" - Etc…

TECHNICAL DESIGN

To be able to build a node with these functionalities we will need the following technical components.



1. A questions queue.
   We have the option to set up a separate Beanstalkd node for this or use on of numerous cloud services available for this.
2. Build an API that can handle all "Simple Questions" , and then iterate further on that by combining series of "simple questions" into "Complex Questions" that we can add to the collection of questions that can be answered.
   This will be custom development
3. Make all our current feeds , and their local cache available to the "Questions queue."
   We already know the feeds , have the code for this and already store a local cache of each feed that we have today. So this is more just making a copy of several databases available to the Questions queue.
4. Build an answers database that the Questions queue can use to store the results in, once a certain question has been answered. (note that all "Complex Questions" will have been saving several answers to "Simple Questions" to be able to construct an answer to "Complex Questions".
   this can be very important performance wise since a lot of the sublevel "Simple Answers" can be

reused in multiple “Complex questions”.
In the previous example for instance , the question “**Which feeds allow searching on VAT number?”** will only need to be asked again when someone has activated or de-activated a certain feed.

5. Build a log database that stores all meta data about each question so that we can use this statistical information in value calculations of datapoints (See later)

So how would this work once we’ve built these 5 subcomponents?

- Let’s keep using the previous example of a simple question.
    - We say we have 2 feeds attached : The Belgian “KBO” and “Open corporates”
    - The KBO feed allows us to download their entire database once a day (since KBO updates their DB once a day) we download their data and store them in the local cache of that feed. Which means that at the start of the “question” , the datapoint we want to retrieve is already stored locally in our systems In the “Feeds” databases
    - The “opencorporates” feed only allows us to submit a specific request for a single VAT number, so that feeds engine is triggered to fire a single API request at a time.
      Since there are way more organisations in the Opencorporates database than we are interested about it is not a sound strategy to “walk” through their entire catalog and pre-fetch this data.
    - We have 1 outstanding question that has been added to the questions queue and for this type of question we already developed a “Answering algorithm”.
        - Question type 5598 = Ask for address base on VAT number
- So this will be our starting point where we already have a pre-cached version of the “KBO” feed , and no results yet out of the “opencorporates” feed since at this point in time no one has asked for one yet.

Questions

Question	Type of Q	Asked by
What is the address of X?	5598	Ecosystem A

Feeds

Source	datapoint	Value
KBO	Address of X	A place
KBO	Name of X	A name

Answers

Question	Answer

- According to the configuration of Question type 5598 these questions can check for answers in both the “KBO” and “Opencorporates” feed so it fires an API call to both those feeds.
    - The “KBO” API first checks it’s local cache and already finds the appropriate answer in its local cache so it simply immediately returns the requested value out of it’s local DB.
    - The “Opencorporate” feed receives the API call , check his local cache and finds nothing so therefor first fires an API call to the “Opencorporates” API, retrieves the result , stores it in his local cache and THEN returns the value which it just stored in its cache. In this case it didn’t find anything (so Opencorporates doesn’t know this VAT number and it also stores and returns this result.
- The 2 API calls that the “Question” of type 5598 needed to go through both returned results
    - KBO feed says address of X = a Place
    - Opencorporates says X = unknown VAT#
- In this case the “Question” is configured to prefer a value over no value so it chooses to return the only value it could find as an answer and saves this result which bring us to this situation

Questions

Question	Type of Q	Asked by
What is the address of X?	5598	Ecosystem A

Feeds

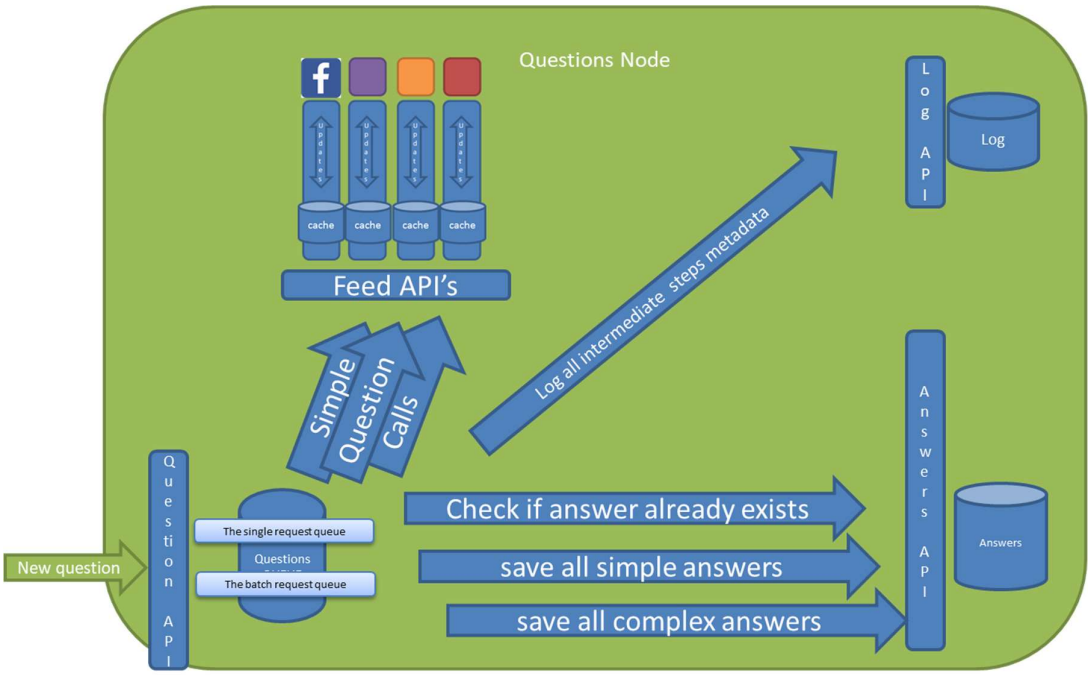
Source	datapoint	Value
KBO	VAT# X	Listed
KBO	Address of X	A place
KBO	Name of X	A name
Opencorporates	VAT# X	unlisted

Answers

Question	Final Answer	KBO answer	Opencorporates answer
What is the address X?	A place	A place	Unlisted

- As soon as a question is answered an outbound call is made to the Core DB API to save this result so that the ecosystem A can now use this newly enriched datapoint in it's reporting model.

So the total sum of types of API calls we'll need to develop will conceptually look like this



While this situation works perfectly well, it requires an immense amount of human configuration and since every feed has it's own specific challenges there are a lot of potential places where we can incorporate AI technology to alleviate this total workload.

Before we start adding AI technologies we will need the following structure of questions.
(Which we already have in our current code repositories)

- Enrich
    - Add datapoints to actors
        - Name available
            - Use name on feed X to retrieve VAT#
            - Use name on feed X to retrieve address
            - Use name on feed X to retrieve url
            - …
        - VAT available
            - Use VAT on feed X to retrieve name#
            - Use VAT to retrieve sector tags from feed Y
        - Url available
            - Use URL to scrape actor website
        - Social handle available
        - Etc…
- Choose
    - Choose feeds
        - Which feeds can this ecosystem use to find VAT#'s
        - Which feeds can this ecostem use to find URL's
        - …
        - Which feeds can this ecosystem use
        - How many values of this datapoint have been filled in by this feed in this ecosystem
        - …
    - Choose actor
        - What is the relevance score of this actor for this ecosystem
        - Which of these two actors should I add to this ecosystem
        - …
    - Choose value
        - Which value should I fill in in this case
        - Which of these tags are most applicable for Actor X in ecosystem A

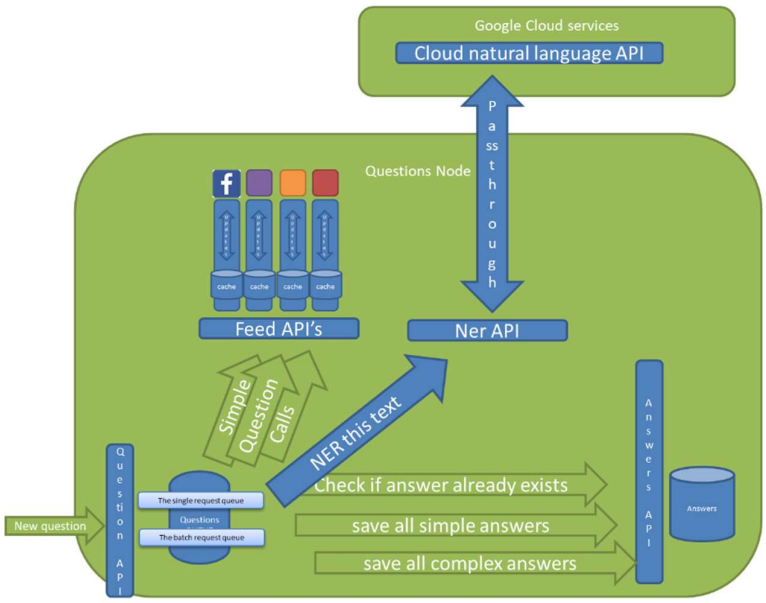
HOW NAMED ENTITY RECOGINITION WILL ENHANCE ALL FEEDS AND IMPROVE QUESTION PROCESSING EFFICIENCY

At this point we do not use "Named Entity Recognition" yet, and using this on all text datapoints in all the different feeds could greatly enhance the types of intelligent insights we could give in our current reporting model.

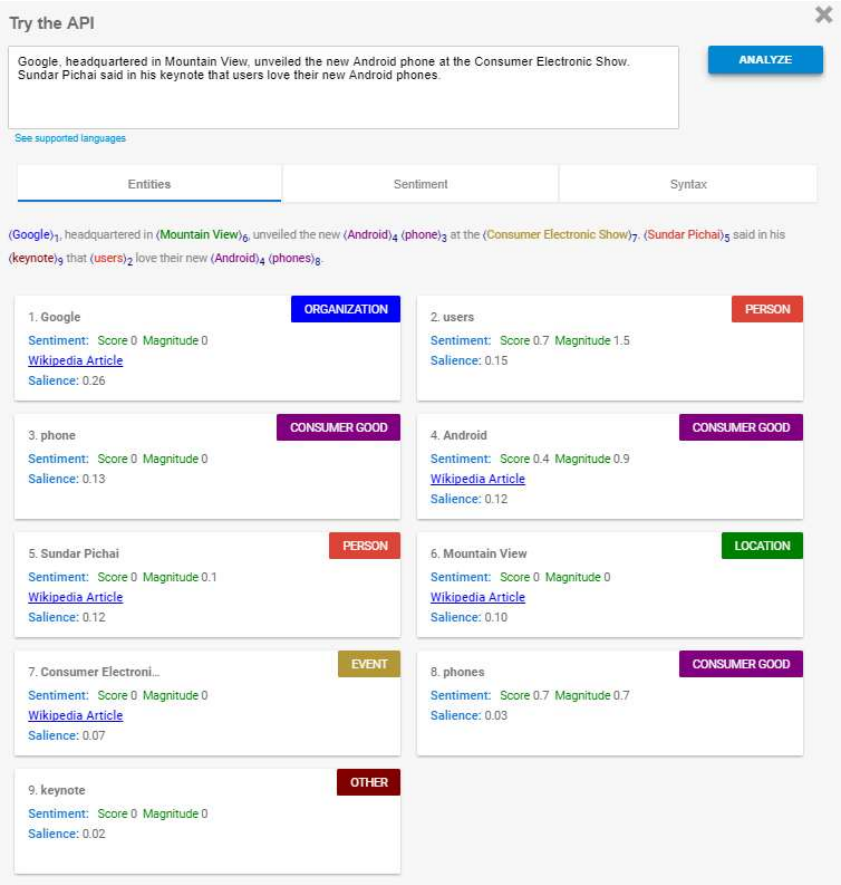
- Structured feeds that gather information about organisations. (KBO , Opencorporates , etc…)
    - Evaluating named entities in descriptions could help us gather information about the relevance of a certain actor, or list of actors we retrieve out of these types of feeds.
      If we for instance ask the "Opencorporates" feed to give us information about the company AFAS in Belgium we get this result.

AFAS *(Belgium, 7 Nov 1995-* , Av du Pont de Luttre 111, Forest, 1190)
AFAS BELGIUM *(Belgium, 13 Aug 1998-* , Schaliënhoevedreef 20, Mechelen, 2800)
nonprofit **Association flémalloise d'Alpinisme et Spéléologie** *(Belgium, 26 Sep 1981-* , Rue Ernest Malvoz 3, Flémalle, 4400) Previously/Alternatively known as AFAS

If the Opencorporates feed could not only store Three new identified organisations in it's local cache, but also store the relative value of the detected entities and their functional relevance it would greatly enhance this particular feed AND the way that we can configure the questions queue.
In this particular case we will add a "NER API" into the "Enhanced Data Feed" node which means we need to do the following things.



- Build an intermediate pass-through API that will allow us to switch out the NER engine actually used at any time. In a first iteration we want to use google's "Cloud natural language API" to get the concept going and once the "Questions node" can use our own NER API we can start setting up our own customised entity recognition engine. (The evaluation of which technology will be done at that point based on the experiences until then)
- Add a new type of "Simple Question" which can be described as "NER this text". The results of such a "Simple Question" return an entire array of data in the following way.



We will log all this information in the Engine's local cache from which the text was gathered, so that we don't need to re-query this expensive external API more than needed.

So this means that as soon as we incorporate our own "NER API" in the node, we can configure the Questions queue's to add this step whenever we see that the returned answer is text. Each feed that we have, now and in the future can start using this NER API , and each feed incorporates the extra metadata in their local cache.

Depending on the questions type we can then choose to include this information in the "Answers" whenever this makes sense.

- There are also a lot of unstructured text feeds
    - A twitter feed (which we already have).
      Each post we find will also be checked for named entities which can then also be stored , linked to that actor.
    - Facebook
      The same for all posts on a companier facebook profile
    - Linkedin
    - A website scraper
      for each url we gather we will also have a bot visit the website and gather all descriptive text from it. (And the structure that those blobs of text are structured in).
      Using the NER API on all those blobs will allow us to extract from them

- A "Forum" scraper
  There are a lot of community sites out there that have a very nice structured overview of topics , and users making posts and answering to them.
  We could add specific forums, for specifis sectors for instance, and start reporting on the evolution of topics and issues being discussed in relation to certain products.
  The forum feed keeps a local cache of all the text data, for each blob the feed check the NER API , and stores the detected entities right next to it so that these results can be directly checked by questions.
  (This can be a companies own internal forum even if they want to incorporate that knowledge into their ecosystem)
- An Email feed
  A company might choose to input all helpdesk or support mails.
  Our mail feed will therefor also use this NER API the same way as the forum scraper would.

- So every piece of text coming into the system will be checked for entities and all will be stored, linked to where it came from and why it was gathered for whom.
  This gives us an enormous pool of data and meta data that we can use to unleash predictive analytics calculations on which will in turn allow us to give intelligent insights about a number of things.

  In practice this means that our DataScientist uses Tensorflow to investigate our data and develop a predictive analytics algorithm in Tensorflow.
  We can test and develop these algorithms on our own dataset in Tensorflow and then immediately run and incorporate the results using Google's "Cloud Machine Learning Engine" .

  Here is a rough overview of the data available for a very simple ecosystem.

Ecosystem	
Basic actor info	name, address, VAT, …
other Enrichment anchors	url, social media accounts , most used hashtags, named entities from own website,
Calculated scores based on known anchors	DS score, social score, SEO score
Feeds	
Twitter	all posts and recognized entities out of those posts
Facebook	all posts and recognized entities out of those posts
KBO	legal company info of all belgian organisation
Opencorporates	Legal company info globally
Crunchbase	allready enriched company info
…	
Questions	a history of all requested updates including metadata (what, why , when, how)
Answers	a history of all found answers including metadata (what , why , when , how)

At this point we've expanded our question repository

- Enrich
- Choose
    -
- Predict
    -

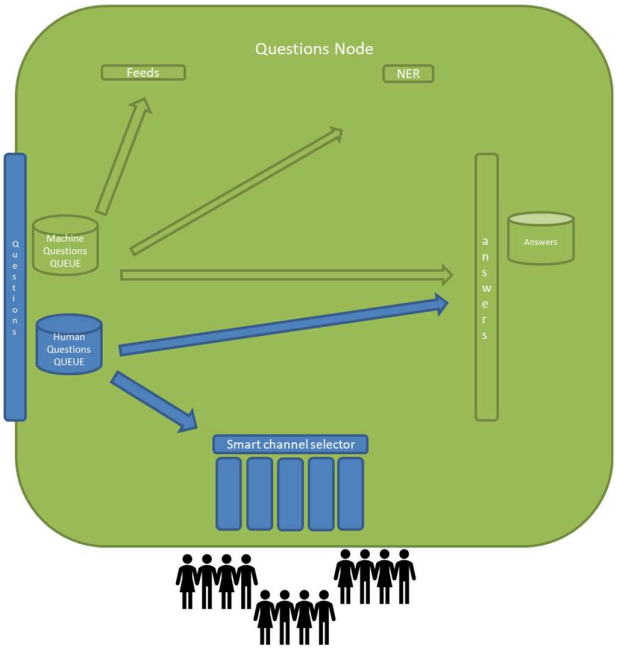
We are expecting to use the results of the data scientists's investigations in the following scenario's

- As recurrent analytical investigations on our users Ecosystems
- As recurrent analytical investigation on our local feed caches
- As extra simple questions
- Enabling "shorter" complex questions

WP2 : HUMAN IN THE LOOP

Once WP1 is concluded we can add the Human feedback in the same technical concept so that we can go from an overall machine learning concept to an overall Active learning concept.

Since we now already have a standardised way to ask questions to feeds , we want to use the same mechanism to ask questions to humans.



The same external API is used (we simply add one parameters that states wether or not the question is meant for humans or automation.

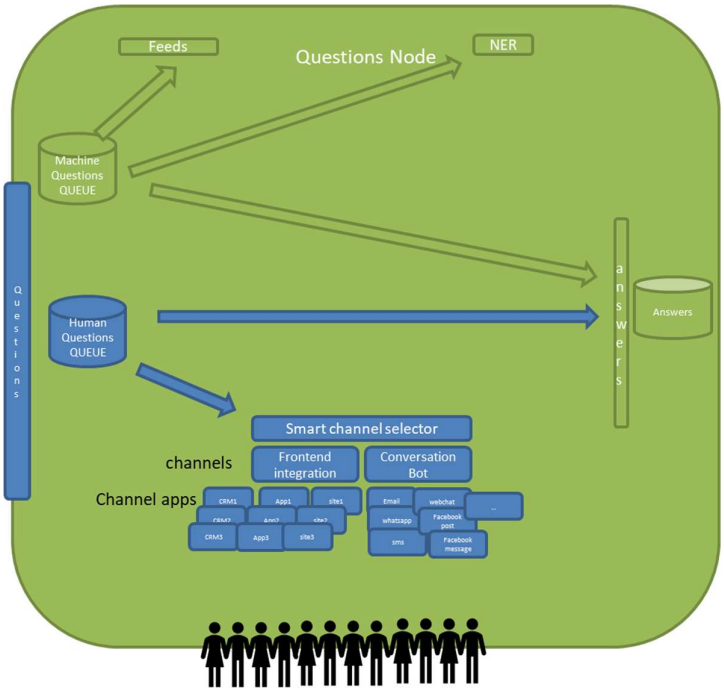
The things we need to build for this are

1. A separate “Human questions queue” since we expect the scheduling and priorities of those questions are totally different. Automated feeds have the potential of generating subsecond answers in a lot of cases, but this will never be so for questions to humans.
2. An API we like to call the “Smart channel Selector” which will eventually be able to choose which “channel” and which “audience” will be the most efficient for a specific questions.
   To explain the function of this “smart channel selector” let’s first explain what “Channels” and “audiences” are.
   - “Channels” are technical means through which a person could be reached.
     We want to give the application the possibilities to interact with humans in very diverse ways and keep thorough statistics about which types of roles and functions of colleagues are more efficient and accurate through which channels for which types of questions and datapoints the minimum list of channels we want to develop are
     - “existing front-end integration” : which basically means we will build an API that can be called upon by any existing front-end application to retrieve questions for a certain user and upload the answers.
       There are numerous examples in which this single API could be used
       1. Our own front-end application. (when we load a certain page we check whether there are outstanding questions for this user and show their number as “notifications”.
       2. Any other web appllications (exisiting internal application at our customer for instance)
       3. Any mobile app could call this api and also generate “notifications” based on these outstanding questions
       4. Any CRM and ERP environment
     - “Conversation Bot” : which basically means we want to engage an existing chat bot implementation and re-use it for different media like
       1. Text chat (skype , facebook , linkedin , slack, whatsapp , …)
       2. Email
       3. Sms

     Obvously these 2 technical channels can be re-used across a wide variety of “channel applications”

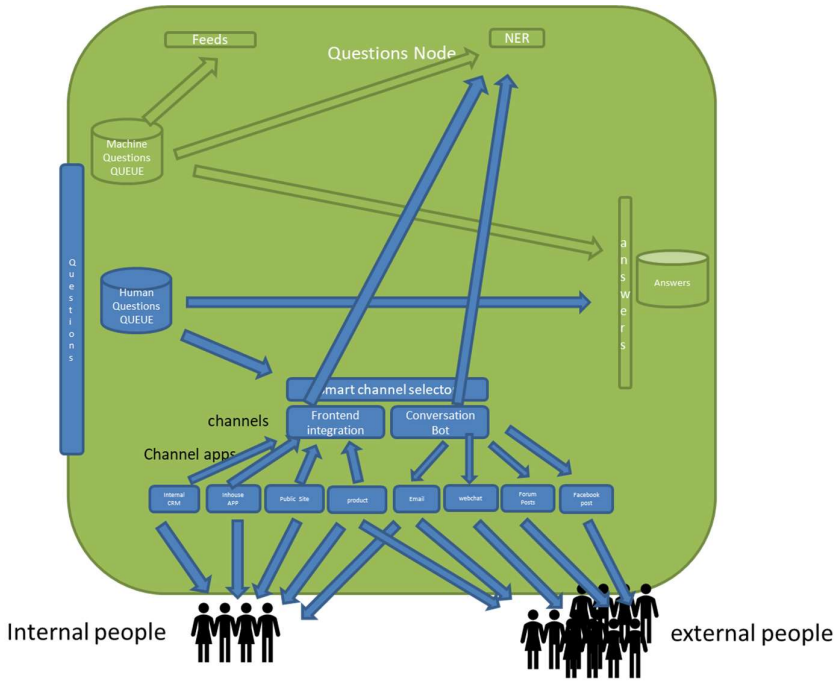
3. And then we have to build and connect the “channel applications” that our customers at that time would prefer to work with. (In which order these will be implemented will be entirely decided by our running customer base at that time.

So we end up with this general situation



Depending on the preferences of our customers this could result in the following operational situation.

We'll describe a potential situation for one single customer.



Let's say one of our customers is an software development company that offers a single product in the form of a locally installable software package.

- Their sales are almost exclusively through their online webshop
- They have an online support website with a forum and ticketing functionality

- They have about 40 internal colleagues (development , marketing , management, …)
- They have a loyal userbase of about 100 000 users globablly

After an initial evaluation of all the options they decide they want to engage the following user groups for the following types of questions.

User groups

- Internal
    - Management experts
    - Data Experts
    - Marketing Experts
    - Other colleagues
- External
    - Beta testing user group
    - Social media followers
    - Existing customer SPOC
    - Anonymous

“Channels” and “channel applications”

- Frontend integration
    - Their own internally developed commercial software product
    - Their own internally developed CRM application
    - Their internally developed mobile app
    - Their public website and more specifically the support and webshop parts of it
- Conversation bot
    - Email
    - Webchat
    - Forum posts
    - Social media posts

And they’ve selected the following matrix in which user groups they want to be reached through which channels

		Frontend integration				Conversation bot			
		Product	CRM	App	Public site	Email	Webchat	Forum posts	Social mec
Internal	Management experts					X			
	Data experts	X	X			X			
	Marketing experts		X			X			
	Other colleagues	X		X		X			
External	Beta testing group				X	X		X	
	Social media followers								
	Existing customer SPOC				X			X	
	anonymous						X		X

It is with this matrix that the Smart composer can start choosing which questions to ask to which sort of human through which channel.

In a first iteration we will manually configure these rules using “Guestimates” and the more questions we start asking, the more statistical information we will have (together will all the other meta data we’ve gathered on datapoints) , we will be able to engage our data scientist to develop AI algorithms to make this “choose” engine more efficient based on the history until then.

WP3 : USER ENGAGEMENT

INTRODUCTION

We now have a lot of technical tools and statistical and meta-data available to us in our backend and as soon as a certain new functionality or metadata is available we can start incorporating it in our software product to engage our users in these tools.

To explain the link between the back-end and front-end work and what we plan to achieve with this we'll start with a timeline.

WP1	Enhancing our datafeed capabi	Months	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
	Build Questions queue																				
		DBA	1	1																	
		Back-end developer	1	1																	
		Data scientist																			
		Front-end developer			1																
	Build Questions API																				
		DBA																			
		Back-end developer	1	1																	
		Data scientist	1	1																	
		Front-end developer			1																
	Build Log DB and API																				
		DBA			1																
		Back-end developer			1																
		Data scientist			1																
		Front-end developer				1															
	Rework Feeds database and API																				
		DBA				1															
		Back-end developer				1															
		Data scientist				1															
		Front-end developer					1														
	Build Answers database and API																				
		DBA					1														
		Back-end developer					1														
		Data scientist					1														
		Front-end developer						1													

WP2	Human in the Loop	Months	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
	Build human questions queue and API																				
		DBA								1											
		Back-end developer								1											
		Data scientist								1											
		Front-end developer									1										
	Build 2 channels																				
		DBA																			
		Back-end developer										1									
		Data scientist										1									
		Front-end developer										1									
	Build smart channel selector API																				
		DBA											1								
		Back-end developer											1								
		Data scientist											1								
		Front-end developer																			
	Build Channel Applications																				
		DBA												1							
		Back-end developer												1							
		Data scientist																			
		Front-end developer												1							

WP3	Use Engagement	months	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
	DataLab																				
		Feed value dashboard						1													
		End-user Feed configurator							1												
		Curation								1											
		Human value dashboard													1						
		End-user Channel configurator														1					
		Enhanced Curation															1				
	Expert Enrichtment																				
		???																			
	Insight																				
		Per datapoint dashboards									1										
		Incorporate new insights in exisiting dashboards										1									
		Human interface dashboard for management experts																1			
		Incorporate human insights in existing dashboards																	1		

		WP1 sum of manmonths	4	4	5	4	4	1	0	0	0	0	0	0	0	0	0	0	0	0	22
		WP2 Sum of manmonths	0	0	0	0	0	0	0	3	1	3	3	3	0	0	0	0	0	0	13
		WP3 Sum of manmonths	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	10
		Total manmonths per month	4	4	5	4	4	2	1	4	2	4	3	3	1	1	1	1	1	0	

DATALAB

EXPERT ENRICHMENT

USING THE CHANNEL API'S IN FRONTEND PRESENTATION

INSIGHTS

WP4 : GDPR COMPLIANCY

AUDIT

DESIGN

IMPLEMENT

WP5 : POC WITH REPRESENTATIVE CUSTOMERS FOR USE CASES