MyCERN

Expert System

Content

Executive summary

An expert system is an indispensable component for MyCERN as it can provide:

1. Intelligent search within all the procedures available at CERN.
2. Description of the rules behind the procedures in a language that is understandable by the system.
3. Suggestions about the most suitable procedures for a certain user and personalized information about them.

The purpose of this document is to evaluate the feasibility of such a system providing all this functionality. It will analyze the different elements of an expert system and the available libraries available in the open source / commercial landscapes.

Finally the document will try to answer to the following questions:

1. Is it possible to build an expert system from the ground up?
2. Are there any open source alternatives out there?
3. Are there any commercial solutions?

Introduction

In the context of computer science, an expert system is a computer system that emulates the decision-making ability of a human expert. Expert systems are examples of knowledge-based systems and traditionally they are composed of two sub-systems: the knowledge base and the inference engine.

The main goal of a knowledge-based system is to make the critical information required for the system to work explicit rather than implicit. In traditional computer programming, all the logic is embedded in the code, making it only understandable by IT specialist. In knowledge-based systems the goal is to specify all these rules in a format that is intuitive and easily understood by business experts.

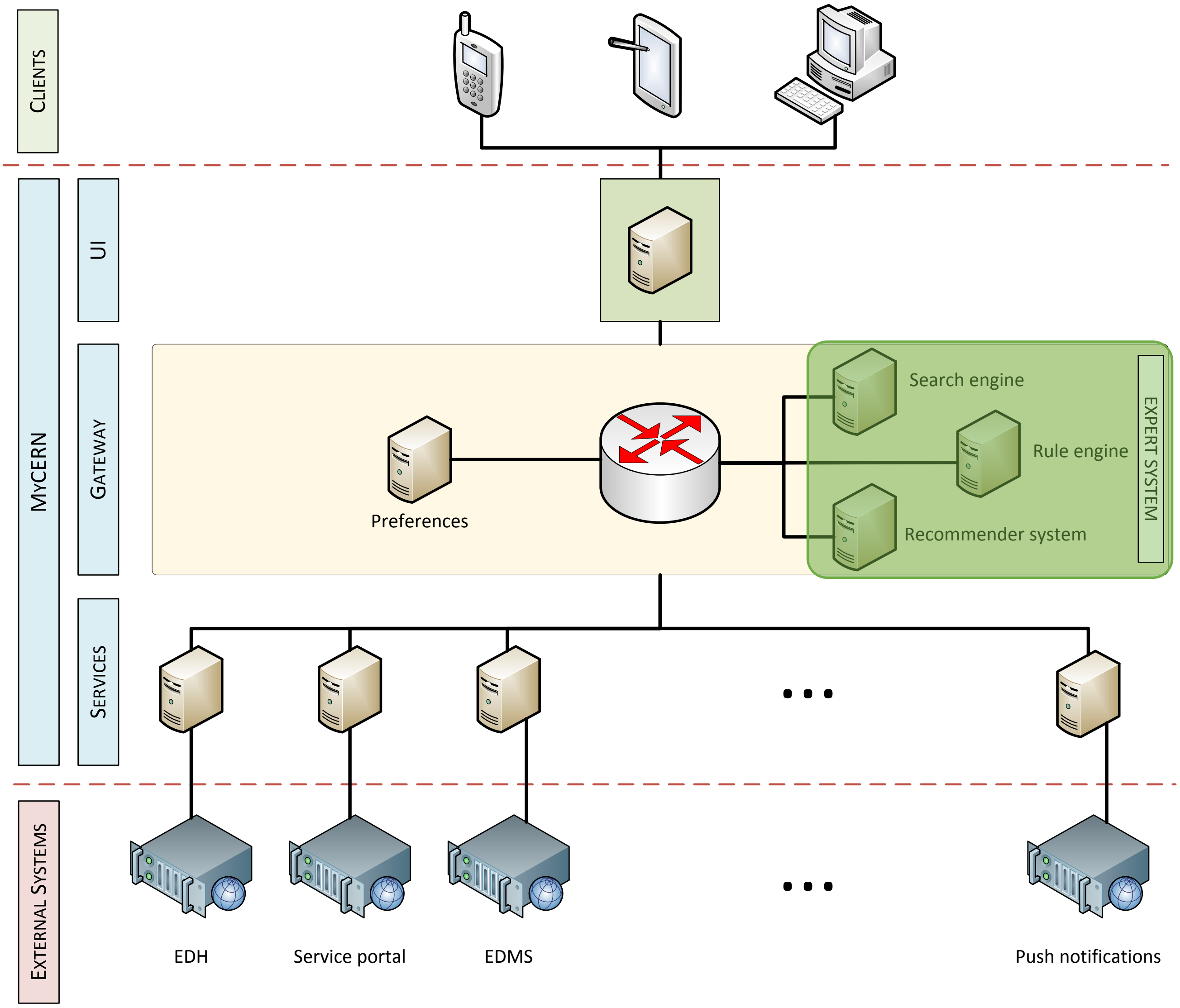
Regarding the MyCERN project, there is a number of key aspects where an expert system can be valuable:

- **Search engine**: Provide useful and precise information in response to user input.
- **Recommender system**: Provide suggestions to the users, tailored to their needs.
- **Rules**: Understanding of CERN procedures and their rules.

Only the rules part is really mandatory as it provides the basic foundations for the MyCERN's knowledge base An intelligent search engine or a recommender system are "nice to have" features that will make the Expert System even more intelligent and valuable but are absolutely not required.

Architecture

In the context of the proposed architecture for MyCERN, the expert system can be decomposed into the three different modules previously discussed:



Search engine

Traditionally, search engines are based on keywords, but it's been proven that keyword search alone is not enough. A search engine needs to understand how the data is related and that's the main reason why Semantic Search has appeared.

A definition given by Wikipedia and Google is the following:

> *Semantic search seeks to improve search accuracy by understanding searcher intent and the contextual meaning of terms as they appear in the searchable dataspace, whether on the Web or within a closed system, to generate more relevant results.*

There are two interesting parts in Semantic Search Engines:

- **Natural Language Processing** (NLP) is a field of computer science, artificial intelligence and computational linguistics concerned with the interactions between computers and human languages.
- **Ontologies**. In computer science and information science, an ontology is a formal naming and definition of the types, properties, and interrelationships of the entities that fundamentally exists for a particular domain of discourse.

Natural Language Processing

The main goal of NLP is to extract structured information from human written text. It can take a user written query an extract a semantic model of the keywords or it can parse a text file (like the ones in the admin e-guide) and fetch important information.

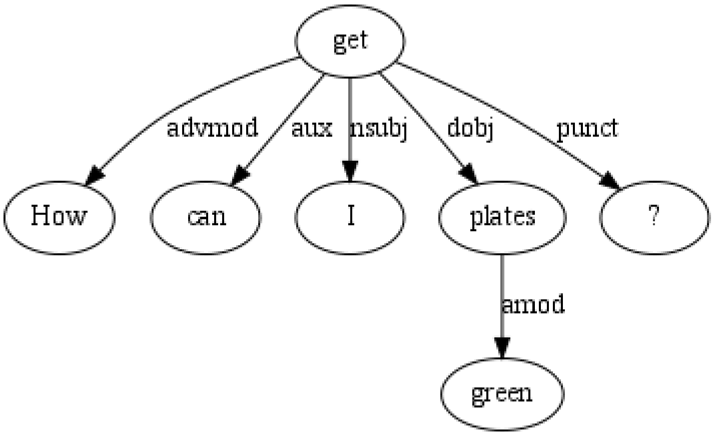
The following example shows where NLP can be helpful for MyCERN. Starting with the following user entered input:

> *How can I get green plates?*

An NLP based solution will tokenize the sentence providing useful information for the search engine to match the query, as an example from the previous query:

Position	Token	Lemma	Part of Speech	Relation to parent
0	How	how	Wh-adverb	advmod
1	can	can	Modal	aux
2	I	i	Personal pronoun	nsubj
3	get	get	Verb, non-3rd person singular present	
4	green	green	Adjective	amod
5	plates	plate	Noun, plural	dobj
6	?	?	.	punct

But I can also provide a graph with the relations between the different words:



There are a number of libraries capable of doing NLP:

1. **Apache OpenNLP**. It is a machine learning based toolkit for the processing of natural language text. It supports all the basic aspects of NLP like tokenization, sentence segmentation … used to build more advanced text processing services.

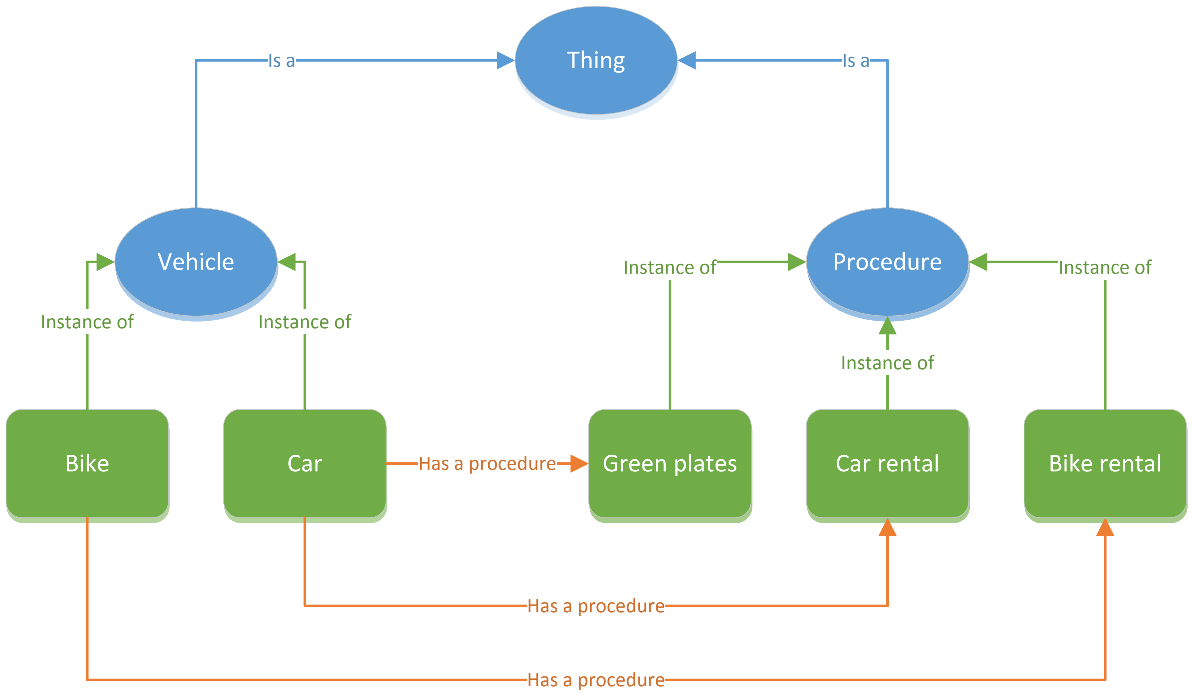
2. **LingPipe.** It is a toolkit for processing text using computational linguistics. It provides the same functionality as OpenNLP but it goes further and has some advance functionality like clustering, database text mining, …

	LingPipe	OpenNLP
License	AGPL: free Developer: $450/server Startup: Starts at $9.500/year/site Enterprise server: $40.000/server	Apache 2.0
Language	Java	Java
Scope	High-level (Off-the-shelf algorithms)	Low-level tools (Lego bricks)
URL	http://alias-i.com/lingpipe/	https://opennlp.apache.org/

Ontologies

The main goal is to provide a representation of the world as we know it, not only the objects but the relationships between them. In the context of MyCERN, an ontology is useful if it can describe the different procedures at CERN and the relations between those procedures and the real-world entities they manage (green plates, French cards …).

The following image shows a small example of an ontology dealing with vehicle related procedures:



Based on the information coming from the NLP analysis plus the information contained in the ontology it is possible to provide useful and precise search results.

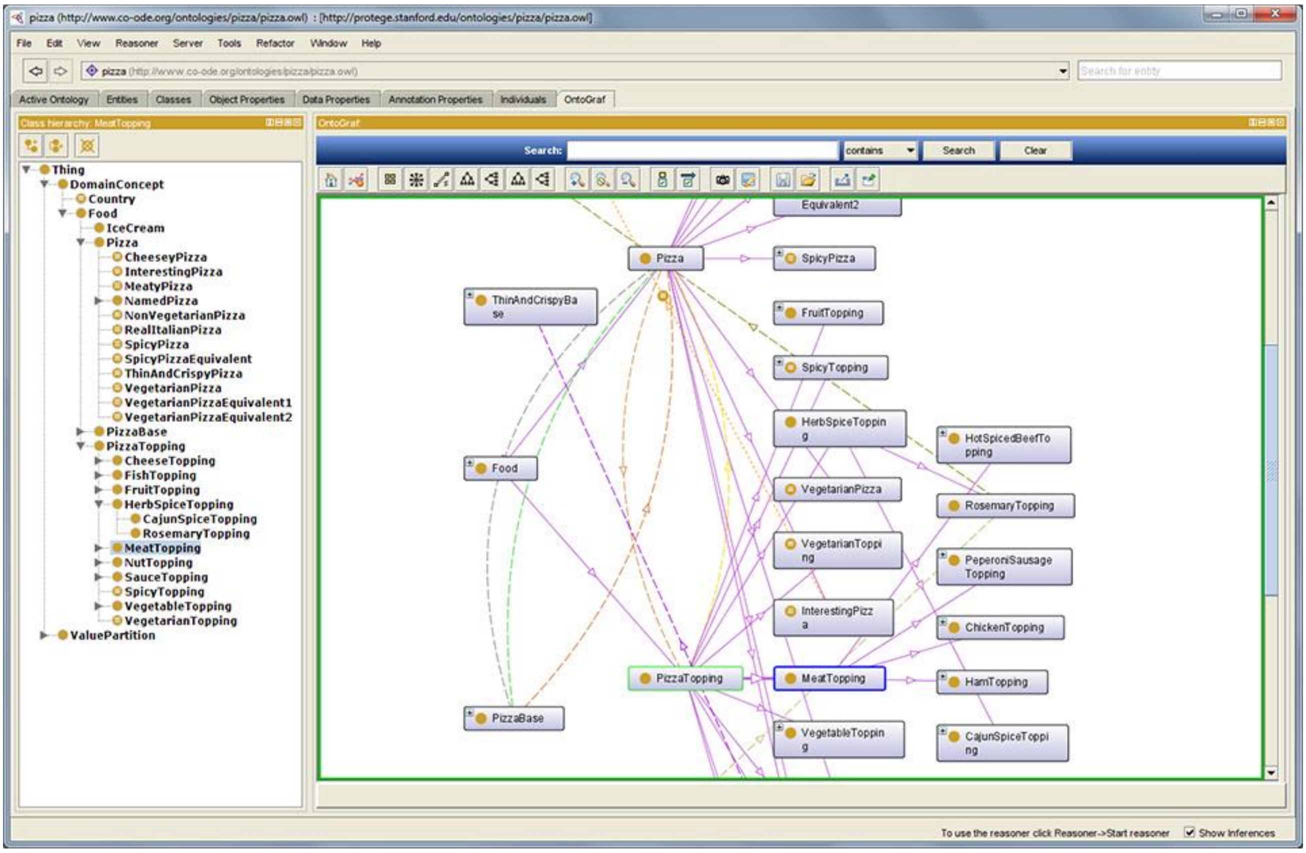
There are lots of tools with the purpose of building/visualizing ontologies:

- **Protégé**. It is a free, open-source ontology editor and framework for building intelligent systems.

- **NeOn Toolkit.** The NeOn toolkit is a state-of-the-art, open source multi-platform ontology engineering environment, which provides comprehensive support for the ontology engineering life-cycle.

	Protégé	NeOn Toolkit
License	BSD	Eclipse public license
Scope	Graphical interface to define ontologies	
URL	http://protege.stanford.edu/	http://neon-toolkit.org/

The following image shows the ontology editor provided by Protégé:

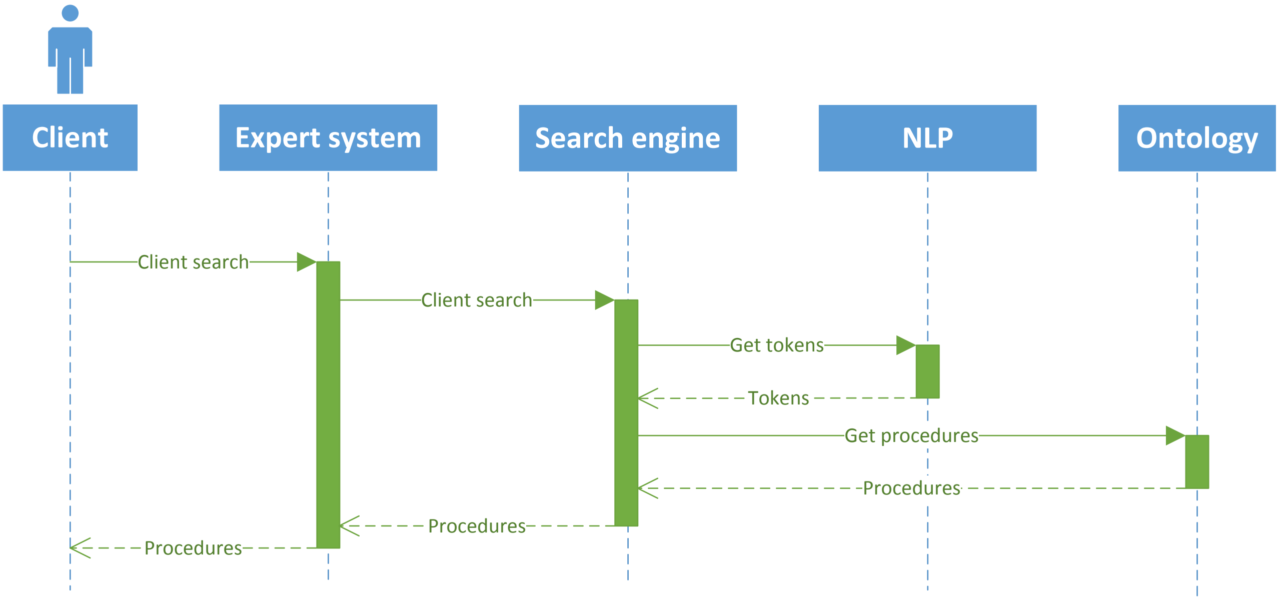


The extra step needed is a language able to query the ontology based on the tokens provided by the NLP analysis (as there is SQL for relational databases). In the case of ontologies, this language already exists and it's called SPARQL.

There is a library called **Apache Jena** that deals with all the internals of saving, writing and querying ontologies.

	Apache Jena
License	Apache 2.0
Language	Java
Scope	Framework for building Semantic Web applications dealing with ontologies and SPARQL
URL	https://jena.apache.org/

The following sequence diagram shows how the search engine will work and its relations with other MyCERN components:



1. The client enters some text into the search box.
2. The expert system forwards the call to the search engine.
3. The search engine calls the NLP module providing the user entered text and receives a list of tokens.
4. The search engine asks the ontology module to perform a SPARQL query based on the tokens in order to find a list of related procedures.
5. The search engine returns the list of procedures to the expert system.
6. The expert system returns the list of procedures to the client.

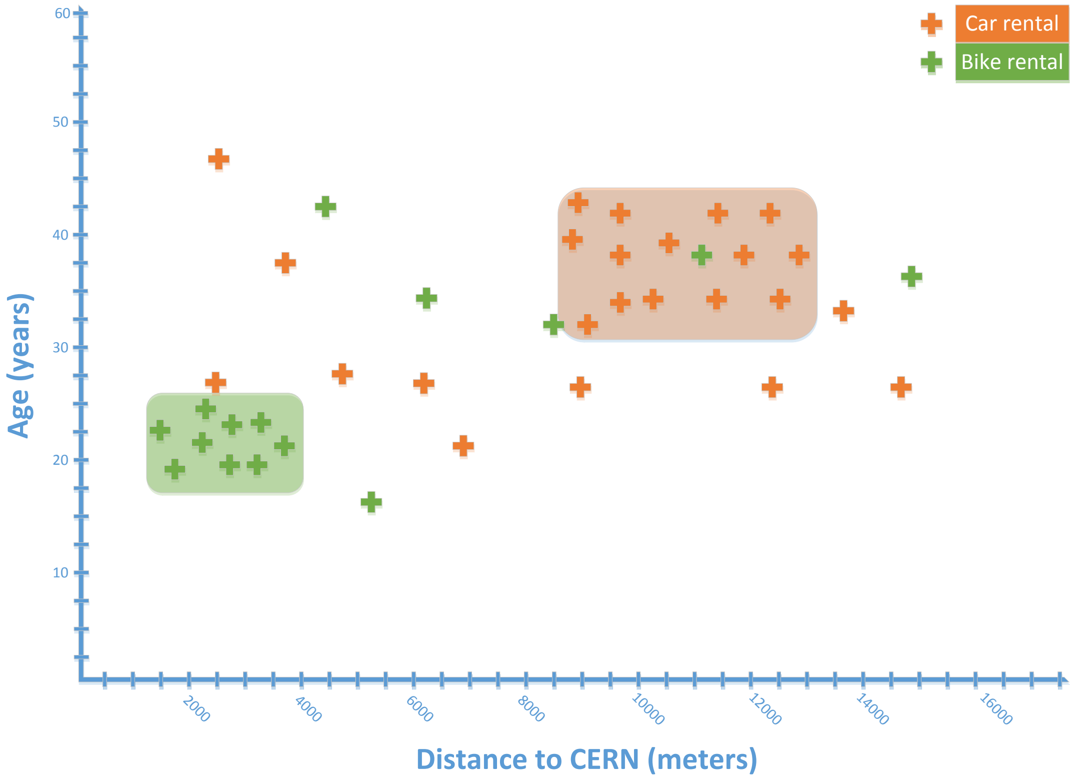
Recommender system

Recommender systems are a subclass of information filtering systems that seek to predict the "rating" or "preference" that a user would give to an item. One of the ways to produce a list of recommendations is to use collaborative filtering that approaches building a model from a user's past behavior (in the case of MyCERN it would be the procedures already initiated by the user or other users with a similar profile).

Building a recommender system for the procedures at CERN can be reduced to a problem of classification/clustering based on historical information regarding those procedures:

- Who created the procedure and when?
- Personal data of the user (age, nationality, address …).

The idea behind is that the system is able to correlate all the information and find patterns/clusters that are not that obvious without a deep analysis, this is the typical problem faced in machine learning and data mining. As an example, for the procedures to rent a car or rent a bike, one can try to correlate the age of the user and the distance from his address to CERN, yielding the following chart as an example:



Each cross will indicate that a user created one procedure of that type. The squares are the representation of the clusters that the algorithm might find (in this case, it's an example of density based clustering), meaning that:

- If the user is between 18 and 25 years old and lives between 1.500m and 4.000m it will be a good candidate to recommend the bike rental procedure.
- If the user is between 30 and 45 years old and lives between 9.000m and 13.000m the car rental procedure will be a good candidate.

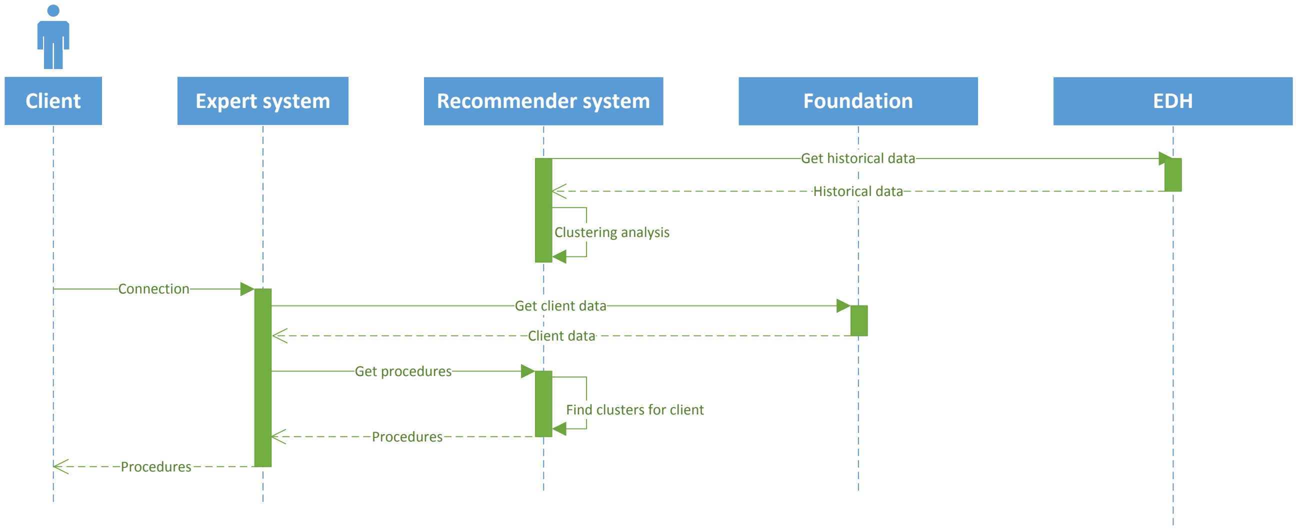
This kind of analysis is not limited to just two axes but it could include as well other variables like the nationality (some nationalities are more used to commuting to work by bike), having or not a driving license …

There are a number of existing libraries that can handle this kind of analysis:

- **TensorFlow**. It is an artificial intelligence/numerical computing library from Google that has been recently released. It provides all the low-level tools to implement a system like the one described before.
- **Scikit-learn**. It is a kit that provides simple and efficient tools for data mining and data analysis. By default, it already implements algorithms to do classification and clustering.

	TensorFlow	Scikit-learn
License	Apache 2.0	BSD
Language	Python	Python
Scope	Low-level tools (Lego bricks)	High-level (Off-the-shelf algorithms)

The following sequence diagram shows how the recommender system will work and its relations with other MyCERN systems:



1. In the background, the recommender system asks to EDH or any other external systems for all the relevant historical data about the procedures initiated at CERN (when, by whom …).
2. Based on that information the system performs the analysis about the different clusters (as stated before, a cluster is just a set of attributes of the client that will make him a good candidate for a procedure).
3. Later on, when a user connects to MyCERN, the expert system asks Foundation or any other external systems for the relevant information about the connected client.
4. The expert system sends the data about the client to the recommender system and asks for a list of procedures.
5. Based on the data about the client and the clustering analysis, the recommender system finds the relevant procedures and returns them.
6. The expert system returns the list of procedures.

Rules

Finally, there is the need to be able to describe the procedures in a language that the computer can understand so he can reason about them. In the end, we want the system to be able to answer to questions like:

> *Is this particular procedure applicable to this user?*

In order to answer this question a mechanism to specify the rules behind the procedures is needed. This is the goal of a rule engine, it is all about providing an alternative computational model to the traditional imperative model, which consists of commands in sequence with conditionals and loops, a rules engine is based on a Production Rule System.

There are lots of rules engines but we can enumerate the following:

- **Drools**. Drools is a business rule management system (BRMS) with a forward and backward chaining inference based rules engine, more correctly known as a production rule system, using an enhanced implementation of the Rete algorithm.

- **Oracle Business Rules**. It is a high performance lightweight business rules product that addresses the requirements for agility, business control, and transparency.
- **IBM Operational Decision Manager**. It is a full-featured platform for capturing, automating and governing frequent, repeatable business decisions.

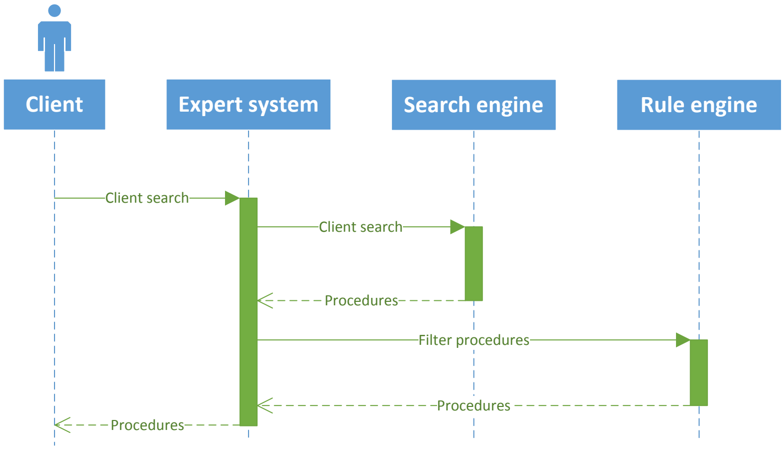
The following example contains the rules needed to know if a user can apply for a French card, the example uses the Drools rules language:

```
package ch.cern.mycern;

rule "Resident in France"
    salience 5
    activation-group "French card"
        when
                who: Person( residenceCountry != "FRANCE" )
        then
                who.setEligible(false);
                who.setEligibleReason("You should get an EF card");
        end

rule "French nationality"
    salience 4
    activation-group "French card"
        when
                who: Person( nationality == "FRANCE" )
        then
                who.setEligible(false);
                who.setEligibleReason("French people don't get French cards");
        end

rule "Titre de sejour prefectoral or resident permanent"
    salience 3
    activation-group "French card"
        when
                who: Person( titreSejour == true || titreResident == true )
        then
                who.setEligible(false);
                who.setEligibleReason("You are the holder of a tritre de sejour prefectoral or a 
titre de resident permanent");
        end

rule "Status at CERN"
    salience 2
    activation-group "French card"
        when
                who: Person( status == "EXTN" || status == "SUMM" || status == "APPR" )
        then
                who.setEligible(false);
                who.setEligibleReason("Your status at CERN doesn't allow you to have a French 
card");
        end

rule "Eligible"
    salience 1
    activation-group "French card"
        when
                who: Person( )
        then
                who.setEligible(true);
        end
```

A rule engine can be useful in several ways:

- Enrich the list of procedures by giving more information about them, like the next steps or why a particular procedure is not applicable to the user.

- Provide a final filter/rank for the results returned by the search engine/recommender system. Maybe a procedure that is suggested is not applicable to a certain user due to his current situation.



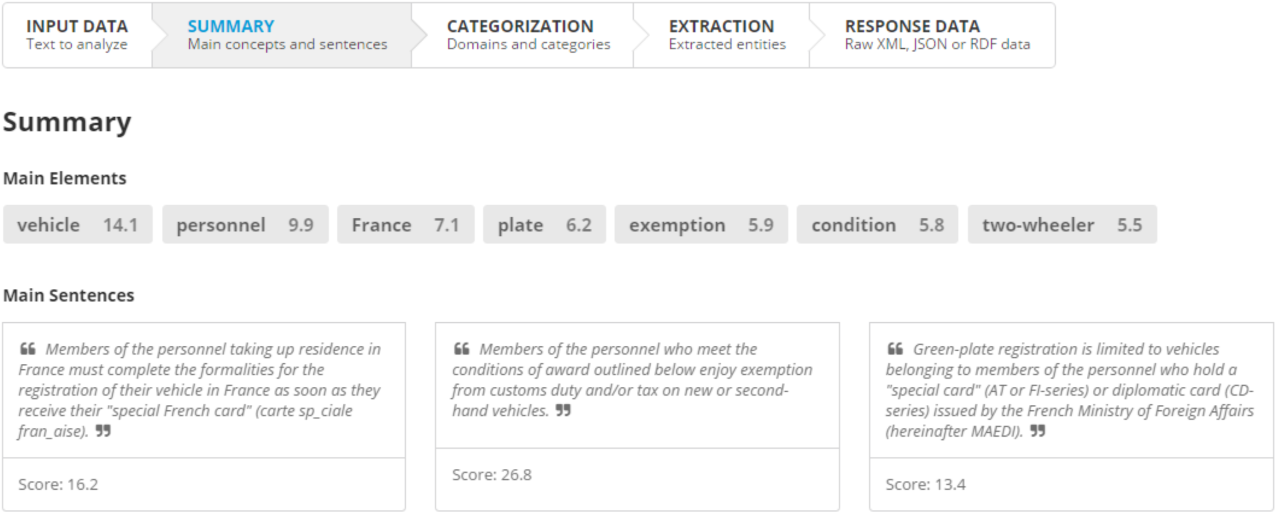
Commercial systems

There is a commercial solution that will fulfill part of the requirements for MyCERN called **cogito** (http://www.expertsystem.com/products/cogito/, that as stated in their website is used by some big companies like Microsoft, Vodafone, ING Direct …

Cogito is a system that:

- It's based on a representation of knowledge called the *Cogito Knowledge Graph* that is something pretty similar to the ontologies discussed previously.
- It uses NLP to understand the words in the same way that people do by using a semantic analysis that is probably quite similar to the one described in this document.
- It is able to learn from human experts and written communications to acquire new knowledge, this is something similar to the clustering analysis.

It will provide all the functionality required by the search engine and the recommender system out of the box. There are a few demos that can be tested online http://www.expertsystem.com/discover-cogito-demo/ to really understand why this commercial system can do. As an example using the first paragraphs from the admin e-guide on green plates, cogito is able to discover the following:



Summary

Main Elements

vehicle	14.1
personnel	9.9
France	7.1
plate	6.2
exemption	5.9
condition	5.8
two-wheeler	5.5

Main Sentences

❝ *Members of the personnel taking up residence in France must complete the formalities for the registration of their vehicle in France as soon as they receive their "special French card" (carte sp_ciale fran_aise).* ❞

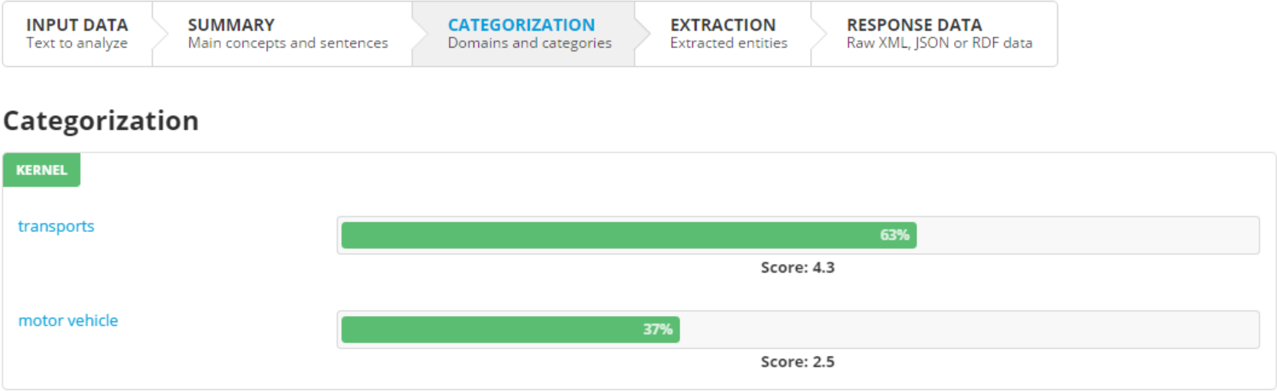
Score: 16.2

❝ *Members of the personnel who meet the conditions of award outlined below enjoy exemption from customs duty and/or tax on new or second-hand vehicles.* ❞

Score: 26.8

❝ *Green-plate registration is limited to vehicles belonging to members of the personnel who hold a "special card" (AT or FI-series) or diplomatic card (CD-series) issued by the French Ministry of Foreign Affairs (hereinafter MAEDI).* ❞

Score: 13.4



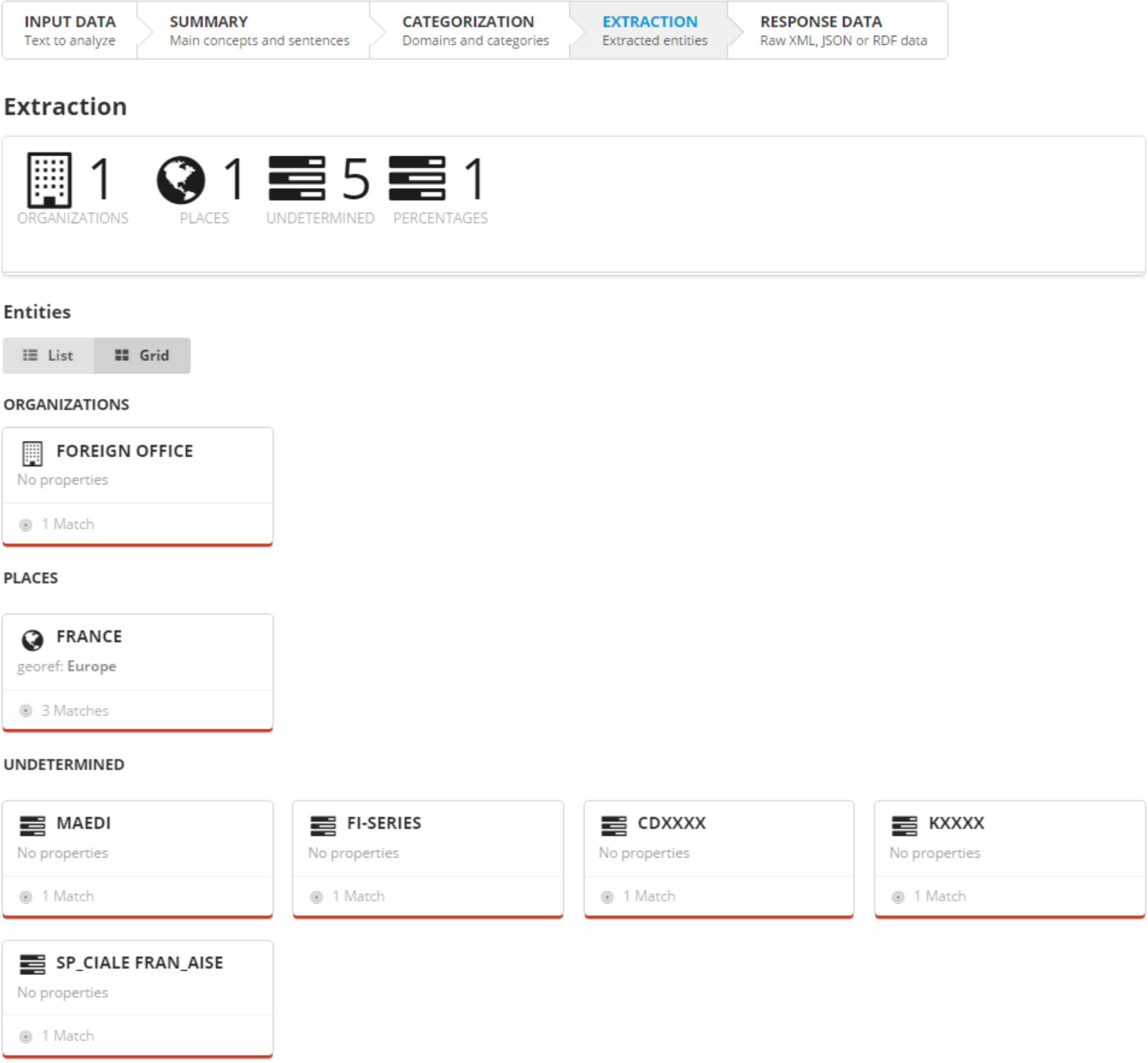
INPUT DATA Text to analyze | SUMMARY Main concepts and sentences | CATEGORIZATION Domains and categories | EXTRACTION Extracted entities | RESPONSE DATA Raw XML, JSON or RDF data

Categorization

KERNEL

transports — 63% — Score: 4.3

motor vehicle — 37% — Score: 2.5



INPUT DATA Text to analyze | SUMMARY Main concepts and sentences | CATEGORIZATION Domains and categories | EXTRACTION Extracted entities | RESPONSE DATA Raw XML, JSON or RDF data

Extraction

1 ORGANIZATIONS · 1 PLACES · 5 UNDETERMINED · 1 PERCENTAGES

Entities

List | Grid

ORGANIZATIONS

FOREIGN OFFICE
No properties
1 Match

PLACES

FRANCE
georef: **Europe**
3 Matches

UNDETERMINED

MAEDI
No properties
1 Match

FI-SERIES
No properties
1 Match

CDXXXX
No properties
1 Match

KXXXX
No properties
1 Match

SP_CIALE FRAN_AISE
No properties
1 Match

So, apart from serving as a search engine or a recommender system, cogito can be used to analyze all the documentation available online in the service portal or the admin e-guide in order to provide valuable information about what is included inside without requiring a business expert to do so.

Conclusion

Is it possible to build an expert system from the ground up?

Yes, this document has listed several open source libraries that can be used to build such a system:

1. By using Semantic Search, it is possible to provide useful results that answer accurately to the questions asked by users.
2. The Recommender System enables the system to identify hidden patterns in regards to procedures that makes it possible to help the user discover those that make more sense to them.
3. Finally, the rule engine can assist in a final step to verify which procedures can be really applied to a certain user.

The following table will show the preferred libraries to be used for each one of the modules of an expert system:

	Option	Reason
Search engine	LingPipe	It provides more functionality out-of-the box, although the license can be a deal breaker
	Protégé	It is one of the most well-known and mature systems
	Apache Jena	
Recommender system	Scikit-learn	It provides a lot of functionality out of the box and it's really simple to use (doesn't require deep technical understanding)
Rules	Drools	It's already used inside CERN so there is a lot of know-how about the tool

Are there any open source alternatives out there?

In the open source landscape, it is possible to find all the required elements to build a system, but so far a sole system that will answer to all the needs of MyCERN has not been found.

Are there any commercial solutions?

Yes, there is a platform called **cogito** that provides solutions to the problems of semantic search/recommender system that should definitely be explored before building an in-house solution.