SEMINAR

2022-04-21

Intro

AI AND DUPLICATE DETECTION TO LEVERAGE EXTERNAL DATA SOURCE

April 21st

Reproductibilité en apprentissage automatique

About myself

ABOUT MYSELF

DAVID BEAUCHEMIN
Ph. D. candidate in ML
Laval University

- B. Sc. in actuarial science
- M. Sc. in computer science, ML and NLP
- 5 years in academic ML and NLP researc project
- +6 years of various applied business projects
- Founder and creator of OpenLayer a bilingual AI podcast
- Founder of .Layer a data science non-profit organisation

À lire

2022-04-21

Reproductibilité en apprentissage automatique

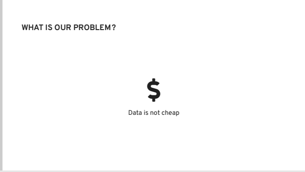
Menu

MENU

à lire

section

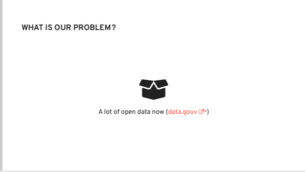
└─What is our problem ?


WHAT IS OUR PROBLEM ?

$

Data is not cheap

You may have heard plenty of times that data is the new oil. But oil, thus data, is not cheap. And we are in a time where we need to move fast ; therefore cannot always properly create new in-house datasets for our needs.


WHAT IS OUR PROBLEM ?

A lot of open data now (data.gouv)

Moreover, with the rise of public datasets curated by large public organizations such as censuses (WHO) and various government, state and cities public datasets, it becomes more and more interesting to use those open datasets.

└─What is our problem ?

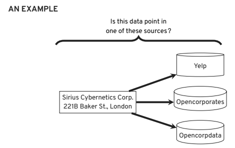
However, a problem arises when trying to use various open data with our private data, that is, mapping data points between different sources.

└─An example

AN EXAMPLE

Only using the name and the address of a business client is it possible to match these with external sources to leverage their content ?

I will use a use case example for the rest of this presentation. It is based on my master thesis work with an insurance company interested in using external sources to lower the number of questions in risk assessments to compute is premiums.


AN EXAMPLE
Is this data point in one of these sources?
Sirius Cybernetics Corp. 221B Baker St., London
Yelp
Opencorporates
Opencorpdata

Let’s use an example to illustrate our problem further. Let’s say we have a client database with business names and addresses, and we would like to leverage external data sources to extract more information about those clients without the need to speak with them.

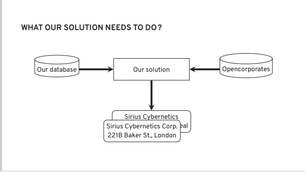
└─An example

A question arises from the example, how can we match data that is not unique ? Names are definitely not unique, and addresses are not ”unique,” but clients can change over time ; thus, a given address can appear multiple times in a dataset.

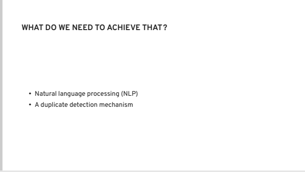
AN EXAMPLE

How can we match our information with external data sources if we do not control it and our information is not unique ?

section


WHAT OUR SOLUTION NEEDS TO DO ?

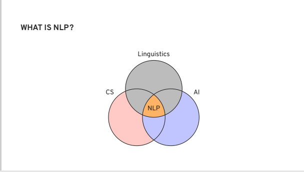
Our solution needs to be able to take at least two databases and extract the duplicate element among the two data sources. Namely, duplicate detection. That is, detecting if two data points are duplicates or not.



WHAT DO WE NEED TO ACHIEVE THAT ?

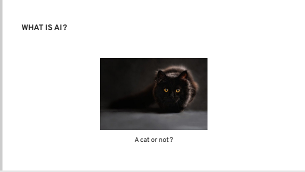
- Natural language processing (NLP)
- A duplicate detection mechanism

To achieve such a solution, we need to use NLP and use or develop a duplicate detection mechanism that will be able to match duplicate records between sources.


WHAT IS NLP ?

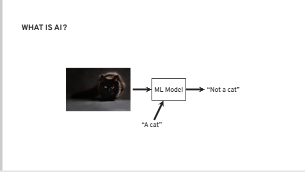
So first, what we need is NLP. So, let's first define what NLP is. NLP is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, particularly how to program computers to process and analyze large amounts of natural language data.

Thus, we need NLP because the data we can use to identify two similar data points will most likely be textual data such as clients' names, addresses, and customer reviews. Otherwise, if we have numerical data such as social security numbers or any numerical data that is a unique mapping to a client, we do not require duplicate detection but rather classical programming rules.

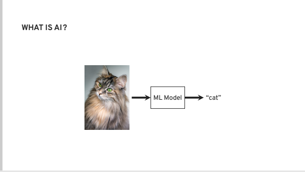

WHAT IS AI ?

A cat or not ?

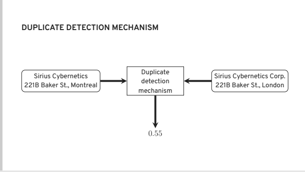
I will not argue about the definition of intelligence or AI. For this presentation, when I say AI, what I mean is machine learning -> ML and, more precisely, supervised ML. The basic idea of ML is to develop a system that can, given data, statistically predict something. For example, let’s say we want to classify if a picture includes a cat. A non-AI system will use heuristic rules such as : does it have fur, claws, etc.


WHAT IS AI ?

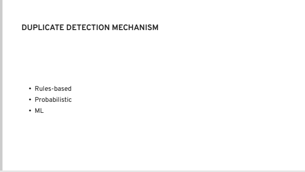
Where an ML approach will use numerous labelled data of cats that include or not a cat and will try to learn his own representation of a cat, where the label are used to give insight to the model where he is mistaken.



and use those internal representations to predict a never seen picture. So, the basic idea of supervised ML is to teach a model or something by giving it examples of the expected output given a context (the image) using labelled examples.


DUPLICATE DETECTION MECHANISM
Sirius Cybernetics 221B Baker St., Montreal
Duplicate detection mechanism
Sirius Cybernetics Corp. 221B Baker St., London
0.55

Our second requirement is a mechanism that can detect if two data points share enough similarity to be considered as a duplicate. Thus, it need to give a sort of a probability of that similarity.

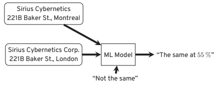


DUPLICATE DETECTION MECHANISM

- Rules-based
- Probabilistic
- ML

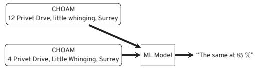
This type of detection is also called record linkage, which is linking records in a data set that refer to the same entity across different data sources. An entity is a business client here composed of the name and address. Such solutions roughly rely on three types of approaches. First, it can use specific rules to determine if two records refer to the same entity. It can also use weighted probabilities of attributes to compute the probability that two records are the same entity. And finally, the approach I will present in the use case in the next part of my presentation is machine learning.

ML DUPLICATE DETECTION MECHANISM



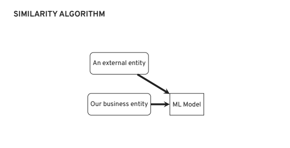
Speaking of the ML approach, using the same ML definition as previously presented with the example with the cat, to build such a system, we need labels and two databases such as illustrated here.

ML DUPLICATE DETECTION MECHANISM


CHOAM
12 Privet Drve, little whinging, Surrey
CHOAM
4 Privet Drive, Little Whinging, Surrey
ML Model
“The same at 85 %”

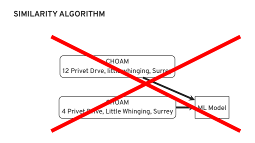
Thus, after training our model, we will be able to take new client information and extract the most probable pair.

section



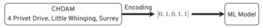
SIMILARITY ALGORITHM

Thus, what we need to build is simply an ML model that will use the textual data of two entities, one from our database and one from an external source.

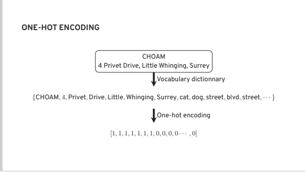

SIMILARITY ALGORITHM
CHOAM
12 Privet Drve, little whinging, Surrey
CHOAM
4 Privet Drive, Little Whinging, Surrey
ML Model

However, a machine learning algorithm needs numbers to work ; thus, they cannot ”simply” use textual data. That is where NLP comes into action.

NLP ML ALGORITHM



So, what we need is to convert (or encode) these textual data into ”numbers” that capture information. There are different ways to do so, each with advantages, disadvantages, and limitations ; I will not go into detail and will only present two.



ONE-HOT ENCODING

$$\begin{tikzcd}
\boxed{\text{CHOAM} \\ \text{4 Privet Drive, Little Whinging, Surrey}} \arrow[d, "\text{Vocabulary dictionnary}"] \\
\{\text{CHOAM}, 4, \text{Privet}, \text{Drive}, \text{Little}, \text{Whinging}, \text{Surrey}, \text{cat}, \text{dog}, \text{street}, \text{blvd}, \text{street}, \cdots\} \arrow[d, "\text{One-hot encoding}"] \\
{[1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0 \cdots, 0]}
\end{tikzcd}$$

An intuitive way to encode textual data is one-hot encoding. That is, creating a reference vocabulary dictionary base on ALL the words that appear in the corpus. Then, using the long sequence of vocabulary, we set all values to 0 and change it to 1 (at a specific index position) if one of the words appears in the sequence. So, using the dictionary in our example, the first seven numbers will be ones, and all the rest will be zeros.

ONE-HOT ENCODING

Advantages

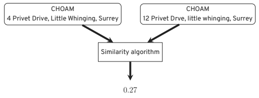
- Easy to implement
- Not complex
- Easy to encode

Disadvantages

- Sparse
- Out-of-vocabulary

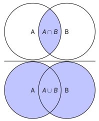
The three significant advantages of this approach are that it is easy to implement, it is not complex to easier to understand for a human, and it is easy to encode sentences using this approach. However, the two major disadvantages are that such representation (encoding) is sparse. That is, they are composed of a lot of zeros. In our example, if we have a cardinality of thousands of words, we will have seven ones and thousands of zeros. For ML models, is it a problem since it has been shown that they tend to not work properly with sparse data. Also, it is impossible to handle vocabulary not seen in the corpus, so new words or unseen words cannot be processed. Thus, we need to find a better way to encode our data.

SIMILARITY ALGORITHM

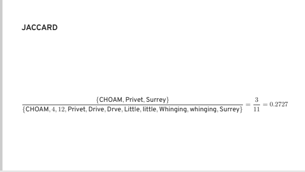

CHOAM
4 Privet Drive, Little Whinging, Surrey
CHOAM
12 Privet Drve, little whinging, Surrey
Similarity algorithm
0.27

Another approach to encoding entities into a numerical value is to use what we call a similarity algorithm. Namely, an algorithm can compute a numerical value based on two strings, as illustrated in this slide.

JACCARD



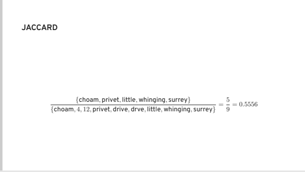
A well know similarity metric is Jaccard. Which is the intersection of tokens (words) between two sentences over all the possible tokens in the two sequences.



JACCARD

$$\frac{\{\text{CHOAM}, \text{Privet}, \text{Surrey}\}}{\{\text{CHOAM}, 4, 12, \text{Privet}, \text{Drive}, \text{Drve}, \text{Little}, \text{little}, \text{Whinging}, \text{whinging}, \text{Surrey}\}} = \frac{3}{11} = 0.2727$$

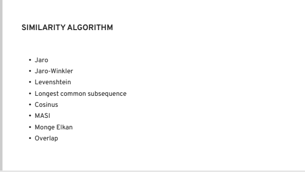
So using our previous example of comparison between two entities will give the following results. Note that this approach is case sensitive, and in that specific case, I did not apply any data cleaning, but it could clearly increase the similarity since two more tokens will be similar (little and whinging).



JACCARD

$$\frac{\{\text{choam}, \text{privet}, \text{little}, \text{whinging}, \text{surrey}\}}{\{\text{choam}, 4, 12, \text{privet}, \text{drive}, \text{drve}, \text{little}, \text{whinging}, \text{surrey}\}} = \frac{5}{9} = 0.5556$$

Note that this approach is case sensitive, and in that specific case, I did not apply any data cleaning, but it could nearly double the similarity since two more tokens will be similar (little and whinging).

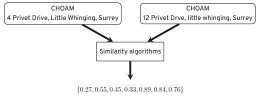


SIMILARITY ALGORITHM

- Jaro
- Jaro-Winkler
- Levenshtein
- Longest common subsequence
- Cosinus
- MASI
- Monge Elkan
- Overlap

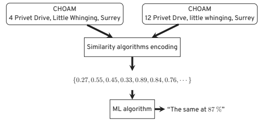
Numerous similarity algorithms exist, such as Jaro, Jaro-Winkler, Jaccard, Levenshtein, overlap coefficient, etc. I will not present all of them since it is out of the scope of this presentation. But there are many different ways to compute the similarity between two strings, where each does not focus on the same thing. For example, Jaro focus on a smaller token unit than Jaccard. Jaro focuses on the character level. So, it becomes interesting to note that computing our example will output different values based on the algorithm used.

SIMILARITY ALGORITHM


CHOAM
4 Privet Drive, Little Whinging, Surrey
CHOAM
12 Privet Drve, little whinging, Surrey
Similarity algorithms
{0.27, 0.55, 0.45, 0.33, 0.89, 0.84, 0.76}

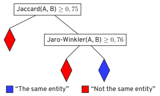
So what I ended up doing in my master thesis was to use numerous similarity algorithms to create a similarity vector between two entities. By using, let’s say, seven similarity algorithms, we can get a vector of dimension seven.

FINAL APPROACH

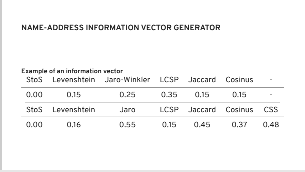

CHOAM
4 Privet Drive, Little Whinging, Surrey
CHOAM
12 Privet Drve, little whinging, Surrey
Similarity algorithms encoding
$\{0.27, 0.55, 0.45, 0.33, 0.89, 0.84, 0.76, \cdots\}$
ML algorithm
"The same at $87\,\%$"

The final approach was something like this. I take two entities and compute a few similarity scores using different similarity algorithms. This vector is then used as the input of an ML algorithm that uses this information to predict using a binary approach (the same or not) along with a given probability.

ML ALGORITHM



Here is an example of an ML model trained using a labelled dataset. We give numerous data examples of an entity composed of a name and address encoded using similarity algorithms and a 0-1 label. One if the entity is the same and 0 otherwise. Then, the training algorithm will find a way to leverage the information vector to classify the comparison.

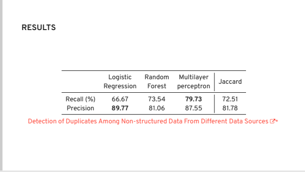


NAME-ADDRESS INFORMATION VECTOR GENERATOR

Example of an information vector

StoS	Levenshtein	Jaro-Winkler	LCSP	Jaccard	Cosinus	-
0.00	0.15	0.25	0.35	0.15	0.15	-
StoS	Levenshtein	Jaro	LCSP	Jaccard	Cosinus	CSS
0.00	0.16	0.55	0.15	0.45	0.37	0.48

Here is an example of a vector computed. The first row is the comparison of the two names, and the second row is the comparison of the two addresses. I did not use the same algorithm depending on whether we compare the name or address. In the end, the vector is composed of 13 elements.

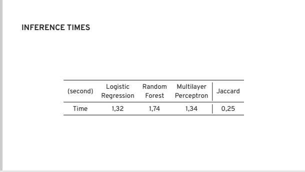


RESULTS

	Logistic Regression	Random Forest	Multilayer perceptron	Jaccard
Recall (%)	66.67	73.54	**79.73**	72.51
Precision	**89.77**	81.06	87.55	81.78

Detection of Duplicates Among Non-structured Data From Different Data Sources

So here are some results of my approach; note that any part of the pre-processing and technical aspect has not been discussed since I focused more on the end goal rather than the complete way to implement it. However, if you have more questions about the details, you can refer to my master thesis (which is in French, but tools like Deepl can convert the part you are interested in) or ask questions personally; I would be glad to respond to more technical questions at the end.
At this point of my work, the baseline is the result of using only the name as the comparison point sing the Jaccard similarity that yields a recall of 72.51. That is, roughly 3 out of 4 business names can be mapped with an external source (in that case, it was the equivalent of open corporate in Quebec). We focus on the recall since we want to match as many entities as possible. We evaluate in second if the precision since, for my problem, a human is still in the loop to validate this; it can reject or accept the proposition.
The random forest and the multilayer perceptron achieve better recall than Jaccard. The best is the perceptron with near 80% recall. So, it means that using both the names and addresses and a complex vector of 14 similarity algorithms only yields a performance increase of around 7%, which is



INFERENCE TIMES

(second)	Logistic Regression	Random Forest	Multilayer Perceptron	Jaccard
Time	1,32	1,74	1,34	0,25

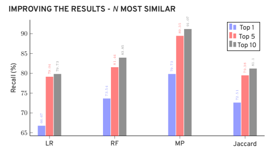
However, at this point, from a business point of view, it can be enough to just use the Jaccard algorithm on the name since it gives quite good results and is less complex and require less computing power, as shown in this slide where ML approach takes more time than only using the baseline approach. Nevertheless, since it was an academic project, I’ve pushed a little further to increase performance.

└─Improving the Results - *N* Most Similar

To improve performance, a straightforward approach is to reduce the task's difficulty. Rather than having an obligation to put the correct answer first, we will consider it a success if the correct answer is in the top *N* most similar (based on the probability). For the top-N, the difference between the first and second was at the fourth or fifth decimal value. Since, in our case, humans are still used in the process, it is not more difficult to select from a top-N rather than accepting or rejecting one element.

IMPROVING THE RESULTS - *N* MOST SIMILAR

We consider a matching is good when the pair *(Our database entity, external data source entity)* is included in the *N* most similar rather than the top-1.



Such an approach gives the most performance increase for all approaches, but mostly on ML algorithm. Thus, we can now at most achieve a recall of more than 90%. However, such an approach did not have a huge impact on precision. Precisions for all approaches are nearly the same.

section

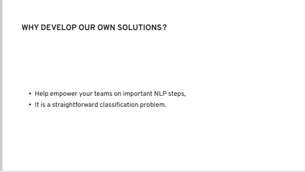
Why develop our own solutions ?

à lire

WHY DEVELOP OUR OWN SOLUTIONS ?

There is commercial solutions out there, why not buying it ?

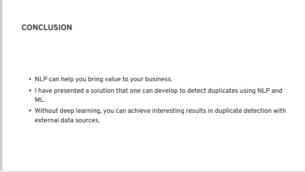
└─Why develop our own solutions ?



WHY DEVELOP OUR OWN SOLUTIONS ?

- Help empower your teams on important NLP steps,
- It is a straightforward classification problem.

À lire



CONCLUSION

- NLP can help you bring value to your business.
- I have presented a solution that one can develop to detect duplicates using NLP and ML.
- Without deep learning, you can achieve interesting results in duplicate detection with external data sources.

conclusion

SEMINAR

2022-04-21

THANK YOU FOR LISTENING !

intro