# Link invalidation project - Information Integration

Daniela Pistol, Sina Akhavan, Mohamed Amine Ben Moussa, Xiaoxuan Hei
January 2020
Master 2 - Data & Knowledge
Télécom Paris, Université Paris-Saclay, Université Paris-sud
Professors: Nathalie Pernelle, Fatiha Sais

***Abstract*— This is report of the project for the course Information Integration. We had 3 different projects to choose from and the project of our choice is the link invalidation project. The aim of this project is to develop a simple tool that helps detect the incorrect sameAs links. Moreover, given two different ontologies, along with the gold standard specifying the correct links between the two, our system will be able to detect incorrect links (if injected to the gold standard) using the functionality degree of the properties. This task is of paramount importance, as is verifying whether a fact is true or not. In a knowledge base, ontology, both the completeness and correctness are important. Hence, the only important thing is not to insert facts or link the KBs, but it's also quite important to verify and check these facts and links. We'll try to explain in details the problem and how we have tried to tackle it.**

## I. Introduction

Linked data is growing everyday as it gives the possibility to semantically query the data. There are billions of triples published as Linked Data to this day. However, when one looks closely, not much of these triples have been linked in between ontologies. Moreover, between these links, there are quite a number of them that are erroneous and thus incorrect links. It is an important task, both to link data between the ontologies and be able to invalidate the ones that are incorrect. For this project we were given the IIMB - Large OWL datatrack [1]. We have used the ontologies 000 and 001 as instructed for **source** and **target** respectively. The gold standard can be found in `refalign.rdf` and we'll describe later how we'll inject some wrong sameAs links. This will allow us to test our invalidation approach as we'll have a ground truth.

[1] http://oaei.ontologymatching.org/2010/im/

This report is structured as follows: In section II, we'll detail how we'll use functional properties for the link invalidation. In section III we'll see how we dealt with objects being represented differently. We used different similarity measures for each of the properties using jaccard similarity, Levenshtein distance, etc. Section V outlines our experiments and reports our results and in the end, section VI will conclude the report.

## II. Functional Properties and Extraction

We have exploited from the definitions and instructions in paper [1]. They have described a property to be **functional** when for each subject, it only takes one object. For instance, consider a knowledge base that only has the 3 following fats: (Tom Cruise, actedIn, Mission Impossible), (Tom Hanks, actedIn, Terminal), (Jessica Alba, actedIn, Sin City). Then based on the definition, actedIn is a functional property. But if we also had the triple (Tom Hanks, actedIn, Forrest Gump), we would have not identified actedIn as a functional property. It is basically not different from the definition of a function so there can not exist two different objects that take part in two facts sharing subjects and properties.

We decided to use a quite different definition for functionality and later use those properties for invalidation. We keep the properties that we find most discriminative by:

1) Computing the number of times each property appears in the knowledge base
2) Computing for each property $p_i$, the number of unique objects that take part in a triple (s, $p_i$, o).

3) We compute the score for each of the properties as follows:

$$score_p = \frac{uniqueObjects}{count_p} \quad (1)$$

For instance take the property *birthYear* and the triples (person1, birthYear, 1994), (person2, birthYear, 1995), (person3, birthYear, 1994), (person4, birthYear, 1993). The count for birthYear is 4 and the number of unique objects for it is 3. Thus giving the score of 3/4.

4) We keep the properties that have a score higher than a threshold as functional properties.

## III. Similarity measures for comparison of properties and objects

### A. property alignments

To compare a fact from the source ontology to a fact from the target ontology, we need to compare both their properties and their objects. We were pleased to see that the properties of the two ontologies have been aligned. For instance the property *directedBy* is the same property potentially with the same domain and range in both ontologies, hence luckily no further work is need to be done with respect to the properties.

### B. Similarity measures

In order to compare the objects (object properties and data properties), it is not enough to use equality. We need to consider different similarity measures. We decided to use different measures to compare the objects based on the properties they hold for. We will first describe the different similarity measures and then explain how we have gone about using them.

*1) Hamming distance:* Hamming distance indicates the number of different bits corresponding to two words with equal length. It means that perform an exclusive-OR operation on two strings and count the number of the result as 1, then this number is the Hamming distance. For example, the Hamming distance between "1011101" and "1001001" is 2. If we index the two words from 0, we can find that the position 2 and 4 are different. With the same method, we can get the Hamming distance between "toned" and "roses" is 3

Hamming distance is very limited for us, because it can only give the distance between two same length words, but in our case, we also need to get the similarity bewteen words with different length.

*2) Levenshtein distance:* The Levenshtein distance is a string metric for measuring the difference between two sequences. The measurement method is to see how many times it takes to process one string into another. For example, the Levenshtein distance between "kitten" and "sitting" is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits:

$kitten \rightarrow sitten$ (substitution of "s" for "k")
$sitten \rightarrow sittin$ (substitution of "i" for "e")
$sittin \rightarrow sitting$ (insertion of "g" at the end)

The Levenshtein distance has several simple upper and lower bounds. These include:

- It is at least the difference of the sizes of the two strings;
- It is at most the length of the longer string;
- It is zero if and only if the strings are equal;
- If the strings are the same size, the Hamming distance is an upper bound on the Levenshtein distance;
- The Levenshtein distance between two strings is no greater than the sum of their Levenshtein distances from a third string (triangle inequality).

*3) Jaro-Winkler distance:* The Jaro similarity of two given strings $s_1$ and $s_2$ is defined as:

$$sim_j = \begin{cases} 0 & m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & otherwise \end{cases}$$

Where:

- $|s_i|$ is the length of $s_i$;
- m is the number of matching characters;
- t is half the number of transpositions.

Only when Two characters from $s_1$ and $s_2$ are the same and there distance is smaller than $\left\lfloor \frac{max(s_1,s_2)}{2} \right\rfloor - 1$, we say they are matching. Comparing the characters matching in s1 and s2, the number of different characters in the same position but different characters is divided by 2 to get t. The

Jaro-Winkler similarity is defined as:

$$sim_w = sim_j + \ell p(1 - sim_j)$$

Where:

- $sim_j$ is the Jaro similarity of $s_1$ and $s_2$;
- $\ell$ is the common prefix length of the string, and the maximum value is 4;
- $p$ is a constant factor, for how much the score is adjusted upwards for having common prefixes. $p$ cannot exceed 0.25, otherwise the similarity will exceed 1. The default value of constant $p$ is 0.1.

The Jaro-Winkler distance is $d_w = 1 - sim_w$.

Let me take 3 words as example, $s_1$ is aboard, $s_2$ is abroad, $s_3$ is aborad. We calculate the JaroWinkler distance between $s_1$ and $s_2$, between $s_1$ and $s_3$. According to Jaro Similarity, the number of matching characters of aboard and abroad is 6, which means $m = 6$. The 3 characters need to to transposed are o, a, r. So $t = \left[\frac{3}{2}\right] = 1$. Their Jaro similarity is

$$sim_{s_1 s2} = \frac{1}{3}\left(\frac{6}{6} + \frac{6}{6} + \frac{6-1}{6}\right) = 0.944$$

For aboard and aborad we can get the same result. It's impossible to determine which is more similar to $s_1$. Now we can calculate Jaro-Winkler distance, provided $p = 0.25$,

$$sim_{ws_1 s2} = 0.944 + 2 \times 0.25(1 - 0.944) = 0.972$$

$$sim_{ws_1 s_3} = 0.944 + 3 \times 0.25(1 - 0.944) = 0.986$$

Now we can conclude that $s_3$ is more similar to $s_2$. However, this method is too complated to use in our project. We want to get the similarity more easily.

*4) Jaccard index:* Comparing with the above two methods of similarity measures, we chose to use Jaccard index. It compares members for two sets to see which members are shared and which are distinct. Its a measure of similarity for the two sets of data, with a range from 0% to 100%. The formula is:

$$J(A, B) = \frac{A \bigcap B}{A \bigcup B}$$

As for comparing two string, we split strings into characters then put characters in sets and then also use the Jaccard similarity to compare.

For example, while comparing "RUB" and "RUeB", $"RUB" \bigcap "RUeB"$ is the number of shared letters and $"RUB" \bigcap "RUeB"$ is the length of "RUB" + the lenght of "RUeB" - the number of shared letters. So the result is 3/4.

*5) Word2vec:* Word2vec is a group of related models used to generate word vectors. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic word text. The network is represented by words, and the input words in adjacent positions need to be guessed. Under the word bag model assumption in word2vec, the order of the words is not important. After training, the word2vec model can be used to map each word to a vector that can be used to represent the relationship between word-to-word, which is the hidden layer of the neural network.

### *C. Object and Data properties*

Another problem that we need to take care of is related to the object properties and data properties. In both ontologies, we deal with the same properties which are expressed by the same value, written in a different manner.

For instance, the **birth date** in one ontology is expressed in the format "1973-08-19", while in the second ontology the date format is the following: "August 19, 1973". A choice we made was to change the format of the second ontology values to the same as the first ontology in order to compare them. Thus, "August 19, 1973" will be changed into "1973-08-19".

Moreover, the **gender** is referred in the source as "Male/ Female" and in the target as "M/F". Using one measure of similarity between these two strings, for instance "Male" and "M" will not give us a very good score. These two strings have just one letter in common but in fact, are referring to the same thing. So, we change the "M" into "Male" and "F" into "Female".

For the property **religion**, we found in both, the source and the target, the same format for the values. The only difference is that, in the second ontology, there are some blank spaces injected between the strings. For instance, "Roman Catholicism", "Agnosticism", "Buddhism" can be found also in the following forms " Roman C at holi-

| Soruce | Target | Jaccard | Soruce | Target | Jaccard |
|---|---|---|---|---|---|
| Constitutional monarchy | CGn1tgtYaionwfjmon!Mczy | 0.3939 | Auckland | 1086.0 | 6886.0 |
| Federation | Fedevation | 0.8181 | Wellington | 290.0 | 560.6 |
| Monarchy | I5U5rc0t | 0.1428 | vancouver_british_columbia | 114.67 | 514.27 |
| Parliamentary system | Par8lampDtary xystUm | 0.6 | | | |

| Soruce | Target | Jaccard |
|---|---|---|
| MYR | MYR | 1.0 |
| RUB | RUeB | 0.75 |
| USD | PUSyD | 0.6 |
| NZD | N4ZfD | 0.6 |
| *USD* | *N4ZfD* | *0.1428* |

cism“ , “Agn ostic i sm “, “B u d d h is m“. We deleted all the spaces we found. Therefore, after applying this change, we compare for equality between the values of the two ontologies.

Considering the **form of the government**, it was quite difficult to find some similarities between the values from the source and the values from target. Even by computing a similarity measures considering the letters of these words, we have very low values, as shown in the following table. Thus, we decided to ignore this property. For Malaysia (item6575592185453392182):

For the **currency** property, we can find either the same currency in the both ontologies, “USD“ - “USD“, either in the target ontology we can find some injected characters between the letters of the currency, “RUB“ - “RueB“. Broadly, we obtain a Jaccard similarity between two similar strings greater or equal to 0.6, as shown in the following table. For two currencies that are completely different, we obtain a very small similarity measure, such as in the last line of the following table.

For properties such as **“actedBy“**, **“directedBy“**, **“spokenIn“** our approach is to compare the number of such properties for each individual. The problem with these properties is that the URI is referred to another object, and not to a value.Thus, we keep only the pairs of subjects from both ontologies which have the same number of such a property.

For the property **name**, we have two cases to study. The first one consists in comparing the names which are the same in the both ontologies - “Lesotho“ : “Lesotho“, “Leytonstone“ : “Leytonstone“. Apart from the same values, the majority of name in the target are just the initials for the name in the source, followed by a dot - “Luke Skywalker“ : “L. S.“, “New York City“ : “N. Y. C.“. In the later case, we’ll change the first form to the second one and compare them.

**Article** is a property which contains the description of the individual. In this case we will compute a similarity measure of paragraphs. Even if the information is expressed using different phrases, the words are mostly the same.

The **calling code** for both ontologies are in the same form. This property is very useful for identifying the country’s sameAs links. In order to have a sameAs link between two countries, the code should be the same. In this case, we check for equality: for Russia, its 7 in both ontologies; for South Africa, 27; etc.

There is a property called **iso_639_1_code**. This property can be found only for two subjects, and its value is either “ru“, either “en“, which states for “Russian Language“ or “English Language“. We say that if we find this property and if its value its equal in both ontologies, the SameAs link its valid.

With regard to the characteristic **size**, the values are very different. We couldn’t find a threshold in order to establish the validity of two sameAs subjects, so this property will be ignored while computing the comparing score. Some examples are presented in the following table.

For the same reasons, the properties **amount** and **estimated_budget_used** will be ignored.

**Starring_in** can be found only in one individual, more exactly in the “Starwars Episode ii attack of the clones“. Same as for the **dialect** which is only used to describe the English language.

**Has capital** is a property which can be very useful for detecting sameAs links between the same country. In the source ontology, this characteristic is expressed in different manners. Therefore, we can find capitals written as “addis_ababa“, the capital of Ethiopia; “bandar_seri_begawan“ the

capital of "State of Brunei Darussalam", "funafuti" for Tuvalu. Unlike the previous ontology, where the capital value is expressed inside the country individual, in the target ontology, the "has capital" characteristic is just making reference to another object which is of type capital. Thus, we can find names such as "item452952361820291081", "item8684125346748819053". In this second case, we keep in a variable the name expressed like this, and we are searching in the ontology for the individual name with the same value as this variable in order to get the name of this item. Once we have the name, we compare it with the capital name from the source, if it's equality we say it is a possible match. The problem comes while comparing these two strings from different ontologies. In the target, the name can be expressed in different forms, such as "A. A.", "B. s. b.", "Funafuti". We need to take care of all these cases, so either we expressed them as the first letter followed by a dot, either we compare directly the strings which have been previously converted to lower cases letters. Equivalently, we use the same reasoning for **born in**.

All the **Filmed_in** values except of one refer to the English language. Consequently, it won't make a big difference if we use this property while computing the score for our comparing sameAs links or not. The only difference it can make is for the Russian language, it can detect this as a match, but talking about just one film referring to the Russian language and much more referring to the English language, we can skip this characteristic, or otherwise it will detect a match for every Cartesian product match between the English films. Idendically, **shot in** refers to United States for all the individuals that it can be found except one, which refers to Russia.

To resume, there are some properties which are more precise and important than others. The choice we make while choosing these properties will make a real impact on the computation and results of precision and recall. As we mentioned before, properties such as birthdate, religion, gender are by far a better choice than the properties we decided to ignore, such as the form of government, size, amount.

# IV.

# V. Experimental Results

## A. Injection of invalid sameAs links to the gold Standard

In order to conduct our experiments regarding invalidating the wrong sameAs links, we first need to inject some incorrect ones intio the gold standards (`refalign.rdf`). To this end, we first transformed the rdf file to a `.tsv` file by parsing the file using DOM parser. Injecting incorrect links is quite simple. We choose a random subject from the source ontology and another random subject from the target ontology. We check to see if the two subject had previously been linked in the gold standard. If that has not been the case, we add the link as an incorrect link to the end of the refalign file. If that has been the case, then we choose two different random subjects and proceed with the checking. We use a parameter **threshold** which allows us to specify the percentage of erroneous links we aim to inject. If the threshold is 0, no wrong links will be added, and on the other hand, if the threshold is equal to 1, we add as much wrong links as correct links in the gold standard.

## B. Results

Our code has been implemented in Python 3 and it's needed to have `rdflib` package installed to run the code. The output of our code will be a list of pairs where the first element of a pair is an object of the first ontology and the second element of the pair will be an object of the second ontology. In other words, every pair contains an object of the first ontology and its match from the second ontology. The next step is to extract information from the `refalign` file and compare it our results. For more clarity of the effectiveness of our tool, we compute the `F-score`. In this context, the precision and recall will be computed as follow:

$$Precision = \frac{Objects\ correctly\ matched}{All\ matches\ produced\ by\ our\ tool}$$

$$Recall = \frac{Objects\ correctly\ matched}{All\ matches\ in\ refalign}$$

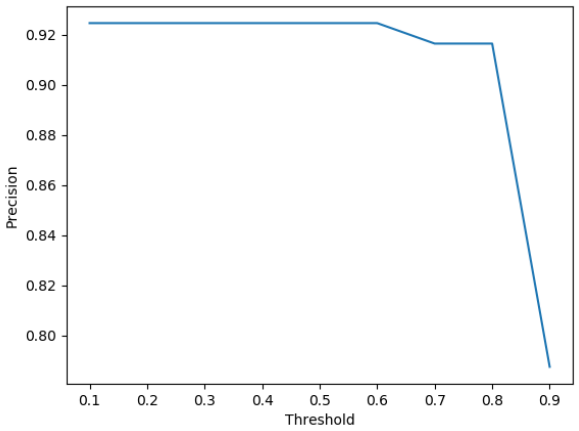


Fig. 1. Precision

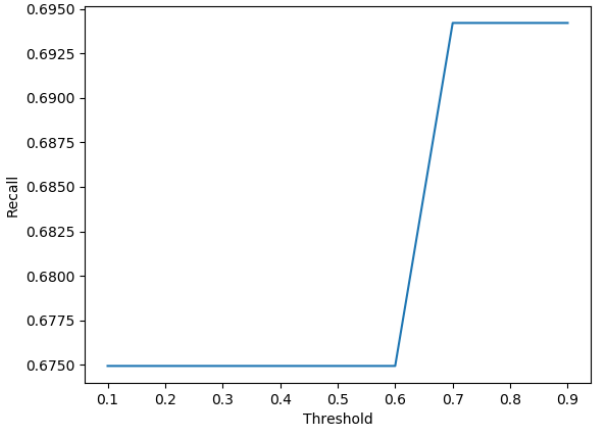


Fig. 2. Recall

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

It is clear that the performance of our tool depends of the properties selected as functional properties for every type. As indicated above, we compute a score for every property. The functional properties will every property having a score above a certain threshold fixed.

Figures 1, 2 and 3 shows the performance results for different thresholds. As seen in fig. 1, the precision decreases when the threshold increases which is totally understandable because when the threshold is low, many properties will be selected as functional properties and that implies that before we can consider two pairs as similar, many properties will be checked and should all have the same value. On the other side, the recall increases when the threshold increases and that is because when the threshold is high, similar pairs are selected easily. That's why fixing an in-between threshold that maximises the F-Score is crucial and according to Fig.3, the best threshold for our two ontologies is 0.7

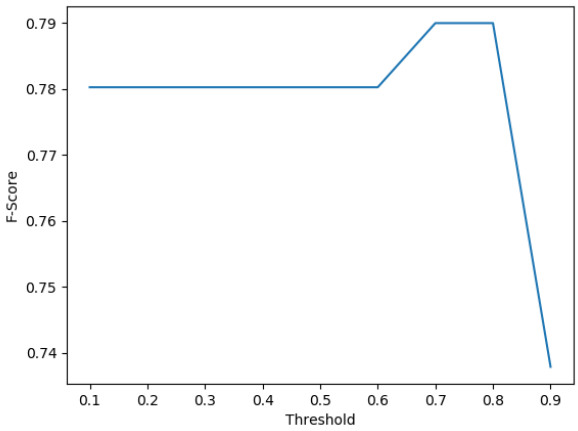


Fig. 3. F-Score

## VI. Conclusion

During this project, we have implemented a way to define the functional properties and concluded that a good selection of functional properties can enhance performance of finding similarities between two different ontologies. Experiments have shown that, with a good threshold, the implemented method gives acceptable performance results.

## References

[1] L. Papaleo, N. Pernelle, F. Sa s, and C. Dumont. Logical Detection of Invalid SameAs State- ments in RDF Data. In proceedings of the 19th International Conference on Knowledge Engineering and Knowledge Management, EKAW 2014, Linkoping, Sweden, Nov. 2014.

[2] J. Raad, W. Beek, F. Harmelen, N. Pernelle, and F. Sais. Detecting Erroneous Identity Links on the Web Using Network Metrics, pages 391407. 09 2018.