EXPERIMENT - 5

AIM: To use linguistic techniques for schema matching.

THEORY:
The fundamental problem is schema matching, which takes two (or more) database schemas to produce a mapping between elements (or attributes) of the two (or more) schemas that correspond semantically to each other. The objective is to merge the schemas into a single global schema. This problem arises in building a global database that comprises several distinct but related databases. One application scenario in a company is that each department has its database about customers and products that are related to the operations of the department. Each database is typically designed independently and possibly by different people to optimize database operations required by the functions of the department. This results in different database schemas in different departments. However, to consolidate the data about customers or company operations across the organization in order to have a more complete understanding of its customers and to better serve them, integration of databases is needed. The integration problem is clearly also important on the Web as we discussed above, where the task is to integrate data from multiple sites.

Schema matching is challenging for many reasons. First of all, schemas of identical concepts may have structural and naming differences. Schemas may model similar but not identical contents, and may use different data models. They may also use similar words for different meanings.

Although it may be possible for some specific applications, in general, it is not possible to fully automate all matches between two schemas because some semantic information that determines the matches between two schemas may not be formally specified or even documented. Thus, any automatic algorithm can only generate candidate matches that the user needs to verify, i.e., accept, reject or change. Furthermore, the user should also be allowed to specify matches for elements that the system is not able to find satisfactory match candidates. Let us see a simple example.

Example 1: Consider two schemas, S1 and S2, representing two customer relations, Cust and Customer.

S_1	S_2
Cust	Customer
CNo	CustID
CompName	Company
FirstName	Contact
LastName	Phone

We can represent the mapping with a similarity relation, over the power sets of S1 and S2, where each pair represents one element of the mapping. For our example schemas, we may obtain

$$\text{Cust.CNo} \cong \text{Customer.CustID}$$
$$\text{Cust.CompName} \cong \text{Customer.Company}$$
$$\{\text{Cust.FirstName, Cust.LastName}\} \cong \text{Customer.Contact}$$

There are various types of matching based on the input information.

1. Schema-level only matching: In this type of matching, only the schema information (e.g. names and data types) is considered. No data instance is available.
2. Domain and instance-level only matching: In this type of match, only instance data and possibly the domain of each attribute are provided. No schema is available. Such cases occur quite frequently on the Web, where we need to match corresponding columns of the hidden schemas.
3. Integrated matching of schema, domain and instance data: In this type of match, both schemas and instance data (possibly domain information) are available. The match algorithm can exploit clues from all of them to perform matching.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA: 3.18)

JUNAID GIRKAR | 60004190057 | BE COMPS A2 | WEB INTELLIGENCE

CODE:

```
from schema_matching import schema_matching

df_pred,df_pred_labels,predicted_pairs =
schema_matching("Table1.json","Table2.json")
print(df_pred)
print(df_pred_labels)
for pair_tuple in predicted_pairs:
    print(pair_tuple)
```

OUTPUT:

```
PS C:\Users\JARVIS\OneDrive - Shri Vile Parle Kelavani Mandal\Desktop\DJSCE\SEM 8\WI\Experiment 5> & C:/Users/JARVIS/AppData/Local/Programs/Python/Pytho
ile Parle Kelavani Mandal/Desktop/DJSCE/SEM 8/WI/Experiment 5/SchemaMatching.py"
schema_matching|Loading sentence transformer, this will take a while...
schema_matching|Done loading sentence transformer
                                data.title  qas.question    qas.id  ...  plausible_answers.text  plausible_answers.answer_start  paragraphs.context
questions.body                    0.002472      0.866227  0.000064  ...                0.001670                        0.000172            0.001018
questions.documents               0.000888      0.011456  0.000316  ...                0.002058                        0.001078            0.000574
questions.ideal_answer            0.000896      0.044691  0.000061  ...                0.001613                        0.003065            0.011124
questions.concepts                0.000594      0.170557  0.000053  ...                0.005908                        0.001418            0.003764
questions.type                    0.004110      0.000979  0.000528  ...                0.000209                        0.001738            0.000112
questions.id                      0.000075      0.000410  0.509553  ...                0.000140                        0.000850            0.000093
snippets.offsetInBeginSection     0.000063      0.000182  0.000119  ...                0.000296                        0.114731            0.000174
snippets.offsetInEndSection       0.000066      0.000222  0.000122  ...                0.000150                        0.177148            0.000165
snippets.text                     0.000282      0.027830  0.000075  ...                0.004836                        0.000167            0.016571
snippets.beginSection             0.001831      0.000836  0.000527  ...                0.000538                        0.000410            0.000513
snippets.document                 0.000643      0.000723  0.000742  ...                0.001803                        0.000477            0.000653
snippets.endSection               0.004702      0.001005  0.000544  ...                0.000472                        0.000416            0.000530
triples.p                         0.000357      0.001243  0.000193  ...                0.009006                        0.000638            0.000383
triples.s                         0.000438      0.096997  0.000329  ...                0.010918                        0.000735            0.000388
triples.o                         0.000965      0.001303  0.000107  ...                0.021065                        0.000123            0.002000
questions.exact_answer            0.000799      0.002843  0.000055  ...                0.566765                        0.001542            0.000229

[16 rows x 9 columns]
```

```
                                data.title  qas.question  qas.id  ...  plausible_answers.text  plausible_answers.answer_start  paragraphs.context
questions.body                           0             1       0  ...                      0                               0                   0
questions.documents                      0             0       0  ...                      0                               0                   0
questions.ideal_answer                   0             0       0  ...                      0                               0                   0
questions.concepts                       0             1       0  ...                      0                               0                   0
questions.type                           0             0       0  ...                      0                               0                   0
questions.id                             0             0       1  ...                      0                               0                   0
snippets.offsetInBeginSection            0             0       0  ...                      0                               0                   0
snippets.offsetInEndSection              0             0       0  ...                      0                               0                   0
snippets.text                            0             0       0  ...                      0                               0                   0
snippets.beginSection                    0             0       0  ...                      0                               0                   0
snippets.document                        0             0       0  ...                      0                               0                   0
snippets.endSection                      0             0       0  ...                      0                               0                   0
triples.p                                0             0       0  ...                      0                               0                   0
triples.s                                0             0       0  ...                      0                               0                   0
triples.o                                0             0       0  ...                      0                               0                   0
questions.exact_answer                   0             0       0  ...                      1                               0                   0

[16 rows x 9 columns]
```

```
('questions.body', 'qas.question', 0.86622685)
('questions.concepts', 'qas.question', 0.17055672)
('questions.id', 'qas.id', 0.5095535)
('snippets.offsetInBeginSection', 'answers.answer_start', 0.9288852)
('snippets.offsetInEndSection', 'answers.answer_start', 0.86390895)
('questions.exact_answer', 'answers.text', 0.5319033)
('questions.exact_answer', 'plausible_answers.text', 0.56676453)
```

CONCLUSION:

Schema matching is a method of finding the correspondences between the concepts of different distributed, heterogeneous data sources. It is a useful technique to create a single global dataset from multiple smaller datasets so that a change in the global dataset can reflect in all the subsets.
