How can I have a duplicate of my hotel removed from trivago?

trivago Hotel Manager
10 May 2017 05:34
Follow

A duplication of your property can appear on trivago because we received double information about it from another booking site.

Our Hotelier Care team can easily resolve this technical issue for you, and you can inform them about it by clicking here.

Meanwhile, continue optimizing the fuller of the two hotel profiles with your **Hotel Information**, which includes **Hotel Details, Images, Room Types,** and **Descriptions**.

Photo credit: Trivago

De-duplicate the Duplicate Records from Scratch

Identify similar records, Sparse matrix multiplication

Susan Li Oct 7, 2019 · 4 min read

Online world is full of duplicate listings. In particular, if you are an online travel agency, and you accept different suppliers that provide you information for the same property.

Sometimes the duplicate records are obvious that makes you think: How is it possible?

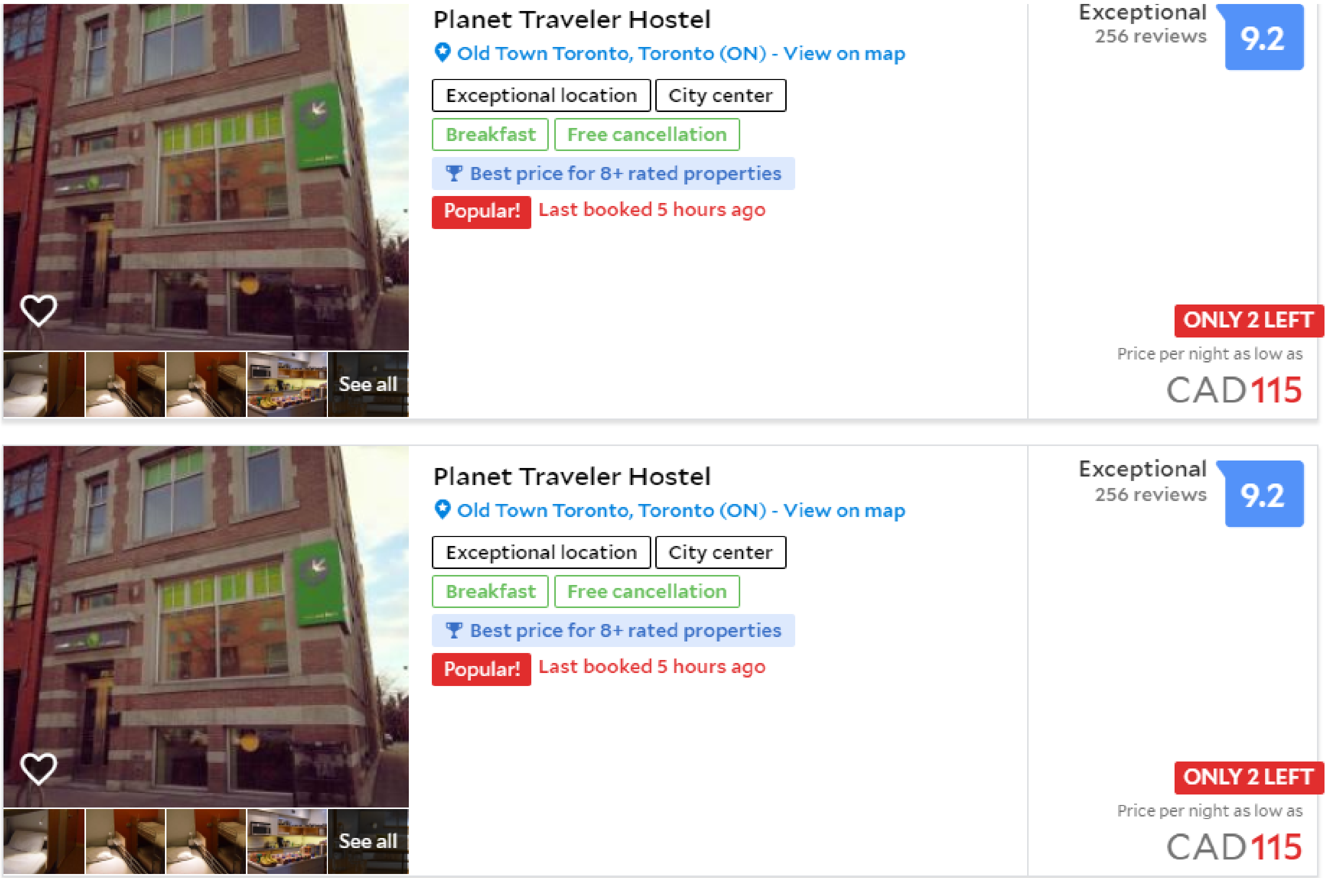
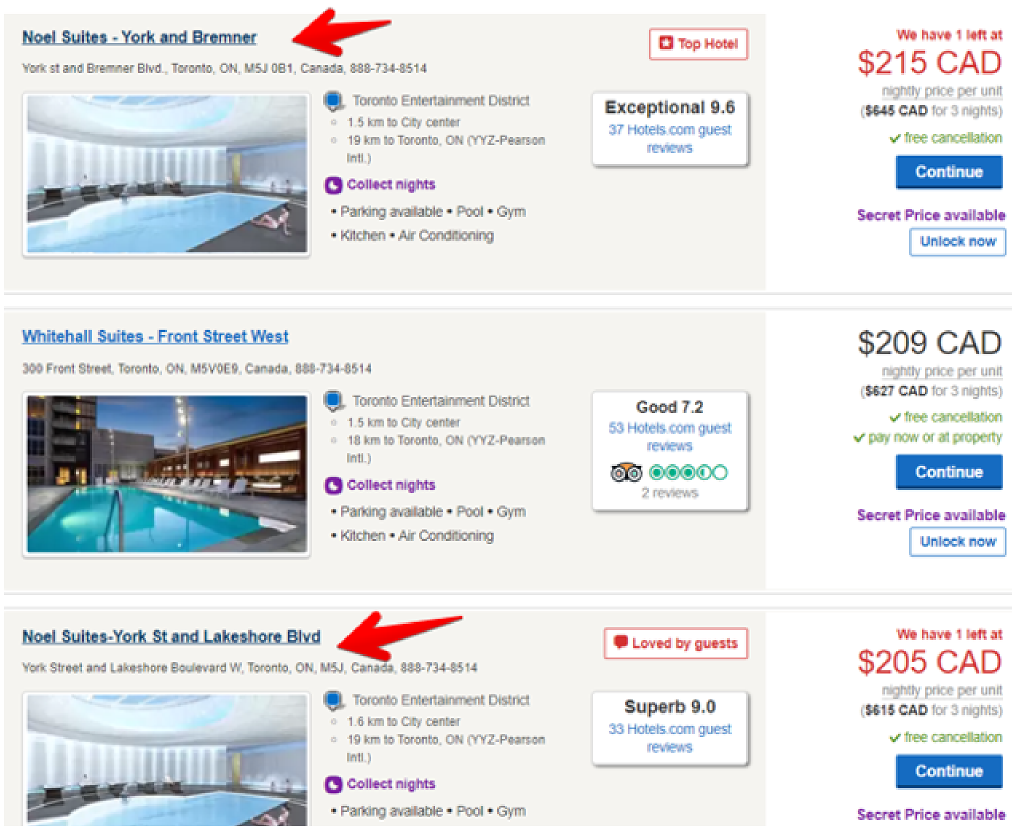


Photo credit: agoda

Another time, the two records look like they are duplicates, but we were not sure.



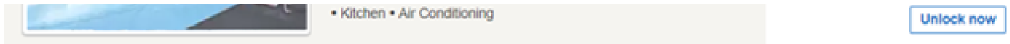


Photo credit: expedia

Or, if you work for a company that has significant amount of data about companies or customers, but because the data comes from different source systems, in which are often written in different ways. Then you will have to deal with duplicate records.

first name	last name	address	phone
`bob`	`roberts`	`1600 pennsylvania ave.`	`555-0123`
`Robert`	`Roberts`	`1600 Pensylvannia Avenue`	

Photo credit: dedupe.io

The Data

I think the best data set is to use my own. Using the Seattle Hotel data set that I created a while ago. I removed hotel description feature, kept hotel name and address features, and added duplicate records purposely, and the data set can be found here.

An example on how two hotels are duplicates:

```python
from scipy.sparse import csr_matrix
import sparse_dot_topn.sparse_dot_topn as ct
from sklearn.feature_extraction.text import TfidfVectorizer

df = pd.read_csv('data/Seattle_Hotels_Duplicates.csv', encoding="latin-1")
df.loc[df['name'] == 'Roy Street Commons']
```

duplicate.py hosted with ❤ by **GitHub** view raw

duplicate.py

	name	address
82	Roy Street Commons	621 12th Ave E, Seattle, WA 98102
90	Roy Street Commons	621 12th Avenue East, Seattle, Washington 98102

Table 1

The most common way of duplication is how the street address is input. Some are using the abbreviations and others are not. For the human reader it is obvious that the above two listings are the same thing. And we will write a program to determine and remove the duplicate records and keep one only.

TF-IDF + N-gram

- We will use name and address for input features.
- We all familiar with tfidf and n-gram methods.
- The result we get is a sparse matrix that each row is a document(name_address), each column is a n-gram. The tfidf score is computed for each n-gram in each document.

```python
df['name_address'] = df['name'] + ' ' + df['address']
name_address = df['name_address']
vectorizer = TfidfVectorizer("char", ngram_range=(1, 4), sublinear_tf=True)
tf_idf_matrix = vectorizer.fit_transform(name_address)
```

tfidf_ngram.py hosted with ❤ by **GitHub** view raw

tfidf_ngram.py

Sparse_dot_topn

I discovered an excellent library that developed by ING Wholesale Banking, sparse_dot_topn which stores only the top N highest matches for each item, and we can choose to show the top similarities above a threshold.

It claims that it provides faster way to perform a sparse matrix multiplication followed by top-n multiplication result selection.

The function takes the following things as input:

- A and B: two CSR matrix
- ntop: n top results

- lower_bound: a threshold that the element of A*B must greater than output

The output is a resulting matrix.

```
def awesome_cossim_top(A, B, ntop, lower_bound=0):

    A = A.tocsr()
    B = B.tocsr()
    M, _ = A.shape
    _, N = B.shape

    idx_dtype = np.int32

    nnz_max = M*ntop

    indptr = np.zeros(M+1, dtype=idx_dtype)
    indices = np.zeros(nnz_max, dtype=idx_dtype)
    data = np.zeros(nnz_max, dtype=A.dtype)

    ct.sparse_dot_topn(
        M, N, np.asarray(A.indptr, dtype=idx_dtype),
        np.asarray(A.indices, dtype=idx_dtype),
        A.data,
        np.asarray(B.indptr, dtype=idx_dtype),
        np.asarray(B.indices, dtype=idx_dtype),
        B.data,
        ntop,
        lower_bound,
        indptr, indices, data)

    return csr_matrix((data,indices,indptr),shape=(M,N))

matches = awesome_cossim_top(tf_idf_matrix, tf_idf_matrix.transpose(), 5)
```

awesome_cossim_top.py hosted with ❤ by **GitHub** view raw

awesome_cossim_top.py

After running the function. The matrix only stores the top 5 most similar hotels.

The following code unpacks the resulting sparse matrix, the result is a table where each hotel will match to every hotel in the data(include itself), and cosine similarity score is computed for each pair.

```python
def get_matches_df(sparse_matrix, name_vector, top=840):
    non_zeros = sparse_matrix.nonzero()

    sparserows = non_zeros[0]
    sparsecols = non_zeros[1]

    if top:
        nr_matches = top
    else:
        nr_matches = sparsecols.size

    left_side = np.empty([nr_matches], dtype=object)
    right_side = np.empty([nr_matches], dtype=object)
    similairity = np.zeros(nr_matches)

    for index in range(0, nr_matches):
        left_side[index] = name_vector[sparserows[index]]
        right_side[index] = name_vector[sparsecols[index]]
        similairity[index] = sparse_matrix.data[index]

    return pd.DataFrame({'left_side': left_side,
                          'right_side': right_side,
                           'similarity': similairity})

matches_df = get_matches_df(matches, name_address)
```

get_matches_df.py hosted with ❤ by GitHub view raw

get_matches_df.py

We are only interested in the top matches except itself. So we are going to visual examine the resulting table sort by similarity scores, in which we determine a threshold a pair is the same property.

```python
matches_df[matches_df['similarity'] < 0.99999].sort_values(by=
['similarity'], ascending=False).head(30)
```

	left_side	right_side	similarity
826	Pike's Place Lux Suites by Barsala 2nd Ave and...	Pike's Place Lux Suites by Barsala 2nd Ave and...	0.715406
831	Pike's Place Lux Suites by Barsala 2nd Ave and...	Pike's Place Lux Suites by Barsala 2nd Ave and...	0.715406
206	Holiday Inn Express & Suites Seattle-City Cent...	Holiday Inn Express & Suites Seattle City Cent...	0.712321

256	Holiday Inn Express & Suites Seattle City Cent...	Holiday Inn Express & Suites Seattle-City Cent...	0.712321
181	Travelodge Seattle by The Space Needle 200 6th...	Travelodge Seattle by The Space Needle 200 6th...	0.669974
211	Travelodge Seattle by The Space Needle 200 6th...	Travelodge Seattle by The Space Needle 200 6th...	0.669974
791	citizenM Seattle South Lake Union hotel 201 We...	citizenM Seattle South Lake Union hotel 201 We...	0.651961
836	citizenM Seattle South Lake Union hotel 201 We...	citizenM Seattle South Lake Union hotel 201 We...	0.651961
586	Quality Inn & Suites Seattle Center 618 John S...	Quality Inn & Suites Seattle Center 618 John S...	0.627400
551	Quality Inn & Suites Seattle Center 618 John S...	Quality Inn & Suites Seattle Center 618 John S...	0.627400
46	Hilton Garden Inn Seattle Downtown 1821 Boren ...	Hilton Garden Inn Seattle Downtown 1821 Boren ...	0.617412
1	Hilton Garden Inn Seattle Downtown 1821 Boren ...	Hilton Garden Inn Seattle Downtown 1821 Boren ...	0.617412
781	Hyatt Regency Lake Washington At SeattleS Sout...	Hyatt Regency Lake Washington At SeattleS Sout...	0.614371
746	Hyatt Regency Lake Washington At SeattleS Sout...	Hyatt Regency Lake Washington At SeattleS Sout...	0.614371
346	Home2 Suites by Hilton Seattle Airport 380 Upl...	Home2 Suites by Hilton Seattle Airport 380 Upl...	0.582787
341	Home2 Suites by Hilton Seattle Airport 380 Upl...	Home2 Suites by Hilton Seattle Airport 380 Upl...	0.582787
561	Renaissance Seattle Hotel 515 Madison Street, ...	Renaissance Seattle Hotel 515 Madison St, Seat...	0.531174

Table 2

I decided my safe bet is to remove any pairs where the similarity score is higher than or equal to 0.50.

```
matches_df[matches_df['similarity'] < 0.50].right_side.nunique()
```

```
152
```

After that, we now have 152 properties left. If you remember, in our original data set, we did have 152 properties.

Jupyter notebook and the dataset can be found on Github. Have a productive week!

References:

https://github.com/ing-bank/sparse_dot_topn/blob/master/sparse_dot_topn/awesome_cossim_topn.py

Super Fast String Matching in Python

Traditional approaches to string matching such as the Jaro-Winkler or Levenshtein distance measure are too slow for…

bergvca.github.io

Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. Take a look.

Get this newsletter

Data Science Nlp Tutorial NLP Naturallanguageprocessing Cosine Similarity