Text dedupe

- Contains python dedupe library
- Dedupe by Record linkage concept

Dedupe 2.0.17

dedupe is a library that uses machine learning to perform de-duplication and entity resolution quickly on structured data.

If you're looking for the documentation for the Dedupe.io Web API, you can find that here: https://apidocs.dedupe.io/

dedupe will help you:

- **remove duplicate entries** from a spreadsheet of names and addresses
- **link a list** with customer information to another with order history, even without unique customer id's
- take a database of campaign contributions and **figure out which ones were made by the same person**, even if the names were entered slightly differently for each record

dedupe takes in human training data and comes up with the best rules for your dataset to quickly and automatically find similar records, even with very large databases.

Facility data examples

Included here are some examples of these issues taken from real, public supplier lists. Some supplier lists are available only as PDF files and processing them with text extraction tools can introduce errors. The two rows in the following table reference the same facility but the text extraction errors make the names difficult to compare.

Name	Address
V .. FR AAS G M BH	ORTER STR. 6, 93233
VFR AAS GMBH	ORTER STR, 6, 93233, GERMANY

All of these rows in the next table refer to the same facility. Note all the variations in punctuation and spelling, and how values within the name and address fields can shift or disappear.

Name	Address
Hela Clothing (Pvt) Ltd. Thihariya	875 Rosimer Estate,Kandy Road,Thihariya,Nittambuwa
Hela Clothing Thihariya	875 Rosimer Estate,Kandy Road,Thihariya, Kalagedihena, Sri Lanka
Hela Clothing (Pvt) Ltd. Thihariya	Hela Clothing (Pvt) Ltd. Thihariya, 875 Rosimer Estate, Kandy Road, Thihariya,Nittambuwa
Hela- Thihariya	No: 875 Rosimer Estate Kandy Road Thihariya
Hela Clothing-Thihariya	No 875, Rosimer Estate, Thihariya, Kalagedihena, Western
Hela Clothing (Pvt) Ltd. – Thihariya	No. 875 Rosimer Estate, Thihariya, Kalagedihena, Nittambuwa, Sri Lanka
HELA C LOTHING THIHARIYA	875 ROSIMER ESTATE,KANDY ROAD,THIHARIYA, KALAGEDIHENA, SRI LANKA

The two rows in the next table refer to two distinct facilities. Some key words in a facility name can be significant (“Washing Plant” in this case).

The two rows in the next table refer to two distinct facilities. Some key words in a facility name can be significant (“Washing Plant” in this case).

Name	Address
Cutting Edge Industries Ltd. (Washing Plant)	1612, South Salna, Salna Bazar Gazipur Sadar, Dhaka
Cutting Edge Industries Ltd.	1612, Dakin Salna, Salna Bazar, Gazipur Sadar Dhaka

Dedupe is a Python library that uses supervised machine learning and statistical techniques to efficiently identify multiple references to the same real-world entity.

Dedupe takes its name from its primary application, looking through a single set of records and attempting to find duplicates. The workflow of the OAR involves comparing a new set of records submitted by a contributor to an existing set of mapped facilities. Dedupe has first-class support for this type of workflow through the use of a gazetteer matcher.

How Dedupe works

We tell Dedupe about the fields in our data we want to compare and the type of each field. “Type” in this context does not refer to a traditional programming language data type but rather how the data in the field should be interpreted and compared. The dedupe documentation includes a list of these types. For the OAR, our field configuration is simple:

How Dedupe works

We tell Dedupe about the fields in our data we want to compare and the type of each field. “Type” in this context does not refer to a traditional programming language data type but rather how the data in the field should be interpreted and compared. The dedupe documentation includes a list of these types. For the OAR, our field configuration is simple:

```
fields = [
    {'field': 'country', 'type': 'Exact'},
    {'field': 'name', 'type': 'String'},
    {'field': 'address', 'type': 'String'},
]
```

Dedupe can match large lists accurately because it uses blocking and active learning to intelligently reduce the amount of work required.

Blocking

Blocking cleverly avoids having to compare each individual record from one data set to each individual record in another by breaking the data into distinct groups.

> Duplicate records almost always share something in common. If we define groups of data that share something and only compare the records in that group, or **block**, then we can dramatically reduce the number of comparisons we will make. If we define these blocks well, then we will make very few comparisons and still have confidence
>
> — *Dedupe documentation*

Dedupe can create two types of blocks, index blocks and predicate blocks. Index blocks group records together by looking up their field values in an index built from the unique values of each field. These lookups are fast, but the index is costly in both construction time and memory size and the blocks it produces are limited to exact matches.

Predicate blocks are more clever and powerful. They are created by grouping records together that share the same features. Here is a simple example from the Dedupe documentation:

Let's take an example. Let's use a "first 3 character" predicate on the **address field** below..

first name	last name	address	phone	record_id
bob	roberts	1600 pennsylvania ave.	555-0123	1
Robert	Roberts	1600 Pensylvannia Avenue		2
steve	Jones	123 Cowabunga Lane	555-0000	3
Stephen	Janes	123 Cawabunga Ln	444-555-0000	4

That leaves us with two blocks - The '160' block, which contains records 1 and 2, and the '123' block, which contains records 3 and 4.

```
{'160' : (1,2) # tuple of record_ids
 '123' : (3,4)
 }
```

Dedupe includes tens of predicate functions that can be combined to create hundreds of different compound blocking rules. Rather than require us to carefully combine predicates by hand in a laborious trial-and-error process, Dedupe, instead, starts by selecting generally useful defaults based on the field types and then uses the active learning process to tune the predicates so they extract the most distinguishing features of our facility names and addresses.

Active learning

The key to any high performing supervised machine learning process is effectively encoding the knowledge of domain experts into training data. Dedupe collects this knowledge using an iterative, real-time training process that emphasizes collecting the most relevant feedback rather than requiring large quantities of labeled data.

Here is an abbreviated example of the interactive training session for our facility data:

```
root@8174a7f28009:/usr/local/src/dedupe# python oar_dedupe_example.py
importing data...
starting interactive labeling...
country : kh
name : grace glory (cambodia) garment ltd
address national road 4, prey kor village, kandal

country : kh
name : grace glory (cambodia) garment ltd
address : preykor village lum hach commune angsnoul district kandal province camb
odia

0/10 positive, 0/10 negative
Do these records refer to the same thing?
(y)es / (n)o / (u)nsure / (f)inished
y
country : cn
name : h u zh o u ti anbao d r e ss c o ltd
address : west of bus station 318 road nanxun town huzhou city zhejiang province

country : cn
name : jiaxing realm garment fashion co. ltd.
address : no. 619 shuanglong road xinfeng town nanhu district jiaxing city
zhejiang 314005

1/10 positive, 1/10 negative
Do these records refer to the same thing?
(y)es / (n)o / (u)nsure / (f)inished
n
```

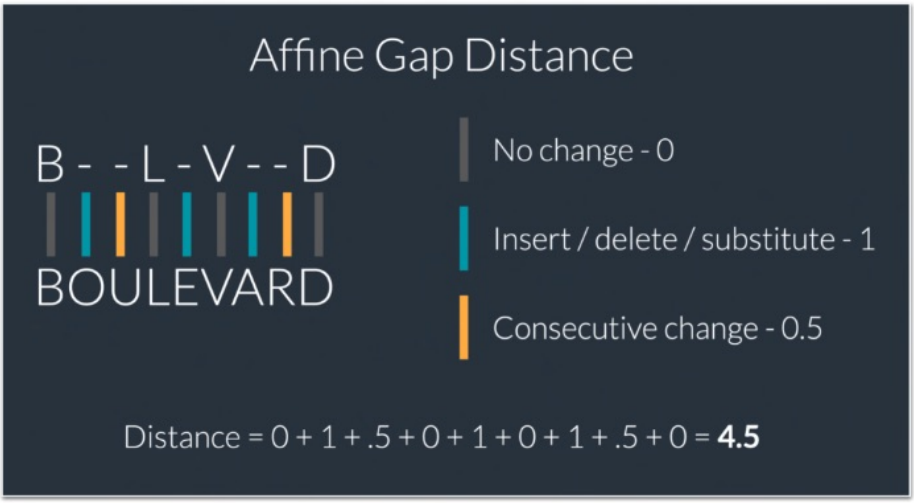
There are a few things to note in this example:
> We are shown both similar and wildly different records. It is important to collect both positive and negative examples.
> The training process can be exited at any time and suggests providing a mere 10 positive and 10 negative matches. Impressively, Dedupe can effectively match data when trained with just this small number of examples. For our production data set we labeled around 50 positive matches and 50 negative matches.

During this training process dedupe is learning two different things about our facility data:

1) How best to convert a measure of the distance between 2 records into a match probability.

2) Which blocking predicates produce the best separation of records into similar groups.

To compare the similarity of our name and address strings, dedupe calculates the affine gap distance between them. Here is an illustration of how the affine gap distance between the strings "BLVD" and "BOULEVARD" is calculated.



Calculating the affine gap distance between two strings

These calculated distances, by themselves, are not valuable on their own. Is “17” good or bad? An important part of dedupe’s active learning process is the conversion of these individual field distance measurements into a percentage probability that the records are a match.

If we supply pairs of records that we label as either being duplicates or distinct, then Dedupe will learn a set of weights such that the record distance can easily be transformed into our best estimate of the probability that a pair of records are duplicates.

— Dedupe documentation

Important links
Documentation: https://docs.dedupe.io/
Repository: https://github.com/dedupeio/dedupe
Issues: https://github.com/dedupeio/dedupe/issues
Mailing list: https://groups.google.com/forum/#!forum/open-source-deduplication
Examples: https://github.com/dedupeio/dedupe-examples

Performing Deduplication with Record Linkage and Supervised Learning

Table of Contents:

What is Record Linkage?

Record Linkage refers to the method of identifying and linking records that correlates with the same entity (Person, Business, Product,….) within one or across several data sources. It searches for possible duplicate records and links them together to be treated as a single record, which also makes it possible to avoid data redundancy.

When unique identifiers variables are present in the data sets such as (Identification numbers, hash codes, etc), the process of linking the same entity will be simple. However, there are cases where unique identifiers are not present in the data sets, therefore we will then need to identify good candidates of variables that are being duplicated and pair them up (eg: State, Last Name, Date of Birth, Phone No) — We will understand more about this while we perform the step: Indexing.

We will be using the Python Record Linkage Toolkit library which provides the tools and functions required for performing record linkage and deduplication. Installation and import of the record linkage toolkit as below:

```
1  pip install recordlinkage
2  import recordlinkage
```

dedup1.py hosted with ❤ by GitHub view raw

Rest read from here https://towardsdatascience.com/performing-deduplication-with-record-linkage-and-supervised-learning-b01a66cc6882