Sonja Lindberg (sonj) & Erica Zhou (ezhou)
7 October 2019
Repo: https://github.com/sonjcl/6.s080datascienceclass.git
Commit Hash: c2f2be18055b50245e51c08920635c84dea6b7b8

6.S080 Lab 3

To begin with, we checked for data completeness and noisiness to determine which features seemed most promising for entity resolution (see image below). **Modelno** looked the most specific and differentiating, so we began by doing an exact match on this field only. We attempted to add brand, but this decreased recall without increasing precision. Therefore, to continue, we matched on TF-IDF (term frequency-inverse document frequency) scores with the vocabulary being the shared words between the two sets of records. Experimenting with different fields, we achieved the best result with a small number of high-value fields from the two datasets. However, one major problem we faced was missing values. For example, 29% of retailer2 records lacked **modelno**. We reasoned that they might have **modelno** hidden in the other fields. We used retailer 1 **modelno** to look in retailer 2 texts, filling in retailer 2 **modelno** with the high TF-IDF scoring find. We got a small further improvement by looking for **modelno** words in the retailer 2 texts and using high-score hits to fill missing **modelno** before the join.

Field1	Retailer1	Retailer2	Field2	Retailer1%	Retailer2%	Uniques1	Uniques2	Comment
RECORDS	850	7365	RECORDS	100%	100%			
custom_id	850	7365	custom_id	100%	100%			
modelno	788	5208	**modelno**	93%	71%	789	5172	clean
brand	803	7364	**brand**	94%	100%	283	1215	rel clean, lower-case and remove space and hypen?
title	850	7365	**title**	100%	100%			
price	850	6459	price	100%	88%	594	3630	
shipweight	780	7218	shipweight	92%	98%	390	513	a x b x c (ounces|pounds)?
shortdescr	647	1412	proddescrshort	76%	19%			wordy, but less than longdescr
dimensions	607	4510	dimensions	71%	61%	572	3630	
groupname	850			100%	0%	51		Coarse category, has & HTML entity
		6936	**pcategory1**	0%	94%		205	Less noisy category
		781	**pcategory2**	0%	11%		130	Less noisy second cat
		6936	category1	0%	94%		545	Noisy category
		781	category2	0%	11%		236	Noisy second cat
longdescr	837	7303	proddescrlong	98%	99%			Wordy description
		7365	asin	0%	100%			Code specific to retailer2
		4874	listprice	0%	66%			905 lack price, of those, 221 have listprice
		6056	techdetails	0%	82%			wordy description
		6422	itemweight	0%	87%			weight
		189	prodfeatures	0%	3%			mostly missing

When we ran our matching code on the training set, we got:

Precision: 0.9169329073482428
Recall: 0.7377892030848329
F1 Score: 0.8176638176638177

When we ran the TF-IDF algorithm, the fields that we found to be most informative were the **category** of the product (*groupname* in *retailer1.csv* and *pcategory1, pcategory2* in *retailer2.csv*), the **title** and **brand** of the product, as well as the **modelno** of the product. We did a small bit of formatting on the **modelno** to remove hyphens because we knew the field was fairly important, and they were inconsistently used in the two datasets. Otherwise, we used the fields as is for TF-IDF scoring since the algorithm works independently of variation like differences in word ordering.

Sonja Lindberg (sonj) & Erica Zhou (ezhou)
7 October 2019
Repo: https://github.com/sonjcl/6.s080datascienceclass.git
Commit Hash: c2f2be18055b50245e51c08920635c84dea6b7b8

In terms of efficiency, we avoided pairwise comparison of all products in a few different manners. In terms of joining data, we only did one direct join on one field (**modelno**), which drastically limited the amount of data that we were comparing, especially since **modelno** is 1 “word” as opposed to fields like the descriptions, which can be much longer. In the TF-IDF algorithm, we also only considered a few fields in building the vocabulary, and again, they were shorter fields to limit the size of the total vocabulary that we were using. Specifically with TF-IDF, we build a joint dictionary between the two datasets to determine word frequency and importance, which involves running through both datasets but not having to directly compare one dataset to the other. Running the algorithm to calculate similarity does require n^2 comparisons, but they use a much smaller subset of data versus making field-by-field/ line-by-line comparisons using all of the data in each dataset.

Feedback:
This was more fun, and the extended timeline made it a lot less stressful and more of an interesting exercise. Very cool!