Data Driven - Assignment 1

Azamat Shora

Nazarbayev University

1 Describe the data.

First of all, we uploaded our data and converted it into the data frame. With a shape of (2600,3). Which means we have 3 attributes and 2600 rows.

Link to the code : link

	Amazon	Google	Match
0	clickart 950 000 - premier image pack (dvd-rom)	clickart 950000 - premier image pack (dvd-rom)	1
1	noah's ark activity center (jewel case ages 3-8)	the beginners bible: noah's ark activity cente...	1
2	peachtree by sage premium accounting for nonpr...	sage (ptree) - vernfp2007rt - premium accounti...	1
3	singing coach unlimited	singing coach unlimited - electronic learning ...	1
4	adobe after effects professional 6.5 upgrade f...	adobe software 22070152 after effects 6.5 pbupgrd	1

Fig. 1. This is the head of our dataframe

The first and second column consists of strings and represents Amazon and Google respectively. The last column is responsible for a Boolean value (1 or 0) which shows the similarity between the previous 2 columns.

The last column is well balanced. This judgment can be made by using ".describe()" method. And the results are next

2 Use algorithms from the String2string library

For each section, I applied each Method and inserted the result into the table.

2.1 Levenshtein distance (at the character level)

2.2 Jaccard distance (at the word level)

2.3 Jaro distance (at the character level)

2.4 Jaro distance (at the word level)

For each section besides the specific method, we also have to pay attention to character level or word level, because depending on this we will apply tokenization or not.

Here are the results of previously described methods. Fig 2.

3 Plot ROC curves for the methods above.

In order to plot the graph I also used the next note.

Note: Jaro and Jaccard distances are inversed similarities (distance = 1 – similarity).

You can see Fig. 3 below.

	Amazon	Google	Match	Levenshtein distance	Jaccard distance	Jaro distance char	Jaro distance word
0	clickart 950 000 - premier image pack (dvd-rom)	clickart 950000 - premier image pack (dvd-rom)	1	1.0	0.333333	0.086802	0.130952
1	noah's ark activity center (jewel case ages 3-8)	the beginners bible: noah's ark activity cente...	1	41.0	0.666667	0.362756	0.430556
2	peachtree by sage premium accounting for nonpr...	sage (ptree) - vernfp2007rt - premium accounti...	1	23.0	0.454545	0.269958	0.216667
3	singing coach unlimited	singing coach unlimited - electronic learning ...	1	31.0	0.571429	0.191358	0.190476
4	adobe after effects professional 6.5 upgrade f...	adobe software 22070152 after effects 6.5 pbupgrd	1	58.0	0.666667	0.35389	0.342857
...	...	...	...	...	...	...	...
2595	shapes	aspyr media inc enemy territory: quake wars	0	39.0	1.0	0.49677	1.0
2596	dragon naturally speaking standard v9	print shop deluxe 21	0	30.0	1.0	0.486729	1.0
2597	mediarecover	sony media software acid pro 5 music productio...	0	46.0	1.0	0.462698	1.0
2598	mediarecover	instant immersion italian 2.0	0	25.0	1.0	0.542146	1.0
2599	photo explosion 3.0	punch software 85100 - punch!master landscape ...	0	58.0	1.0	0.537058	1.0

Fig. 2. We applied all four methods and presented results

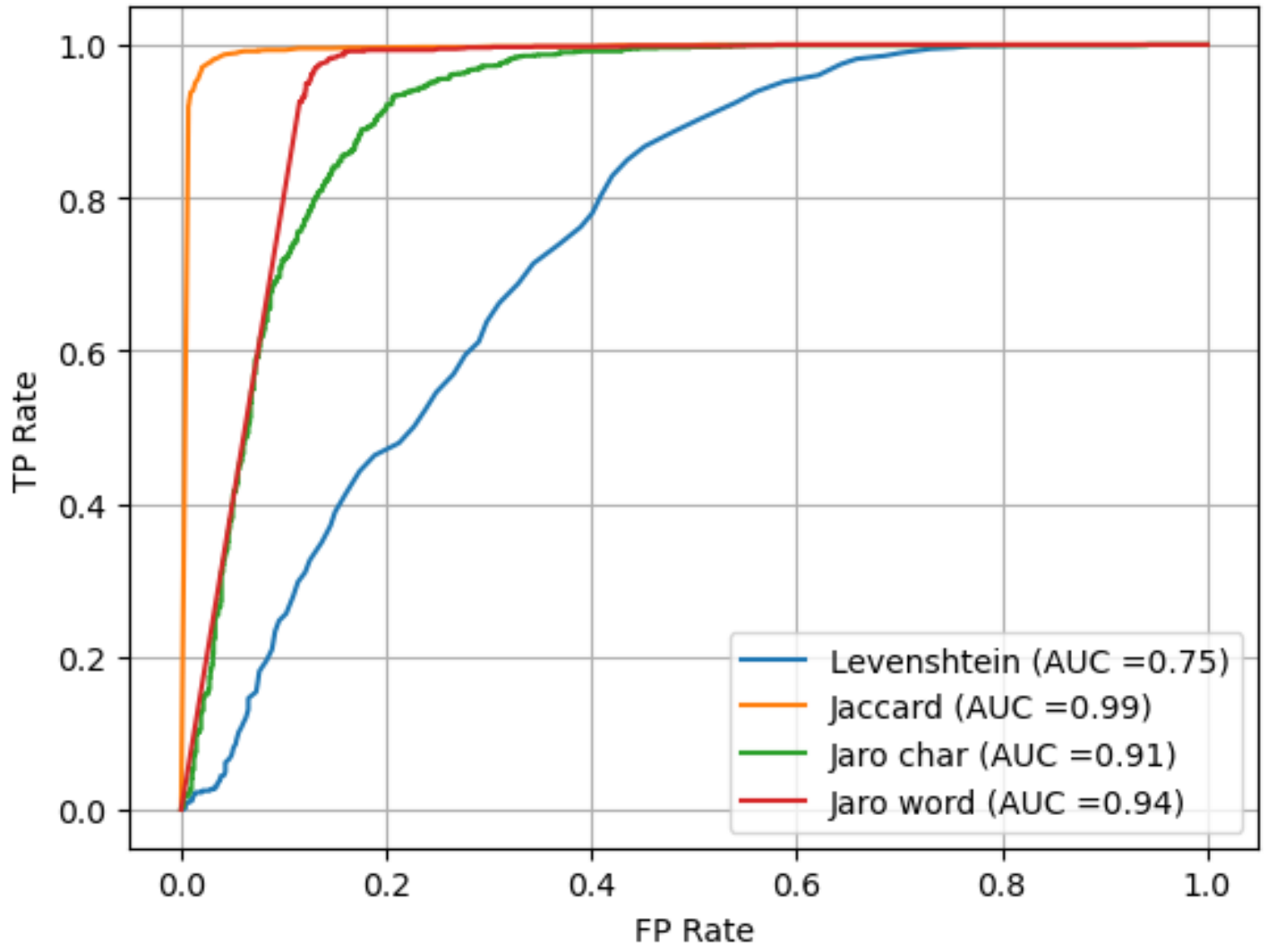

Fig. 3. All the methods with measured AUC

4 Analyze results

Based on this graph we understand that the performance of the Jaccard method (AUC = 0.99) shows the highest performance. Next to Jaccard by performance, we see the Jaro distance method (at the word level) with AUC = 0.94. After that goes the Jaro distance method (at the character level) with AUC = 0.91. And the lowest performance can be noticed in the Levenshtein method with only AUC = 0.95.

5 Propose string pre-processing that can improve results, report results

For string pre-processing, we can apply techniques like:

- Lowercasing: ”Apple” -> ”apple”
- Removing punctuation: ”Education,!+” -> ”Education”
- Stemming: [’run’, ’running’, ’runs’] -> ’run’

By using these functions we will ”denoise” our data and thereby increase the final results.

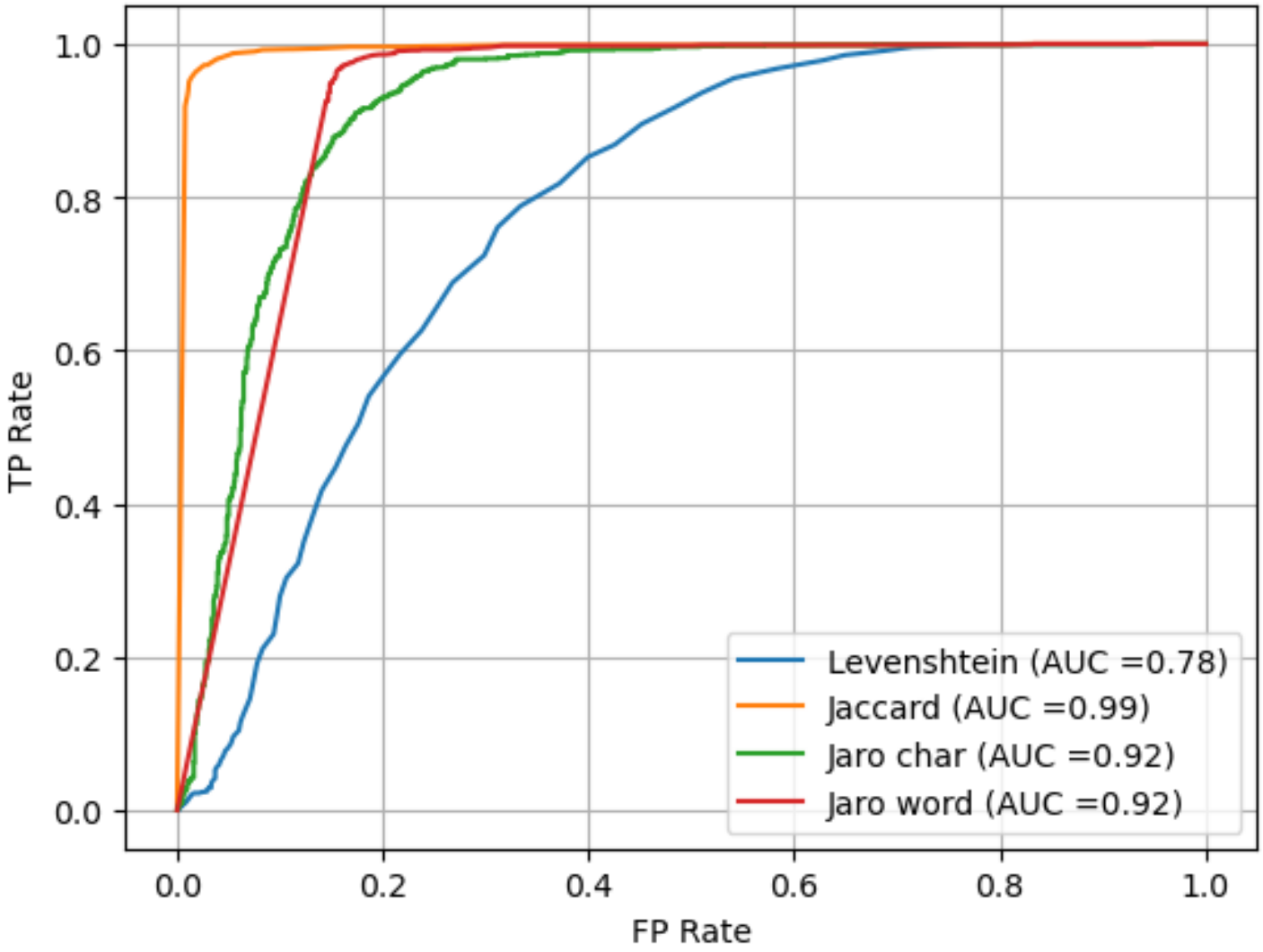

Fig. 4. All the methods with measured AUC, after applying pre-processing

Based on the new results we can make the next Table 1.

	Before pre-processing	After pre-processing
Levenshtein distance (at the character level)	0.75	0.78
Jaccard distance (at the word level)	0.99	0.99
Jaro distance (at the character level)	0.91	0.92
Jaro distance (at the word level)	0.94	0.92

Table 1. A Comparison of AUC before and after pre-processing

Based on this table we can make next points:

- Levenshtein distance (at the character level): results were improved from 0.75 to 0.78
- Jaccard distance (at the word level): results stays the same
- Jaro distance (at the character level): results were slightly improved from 0.91 to 0.92
- Jaro distance (at the word level): results were dropped from 0.94 to 0.92

6 Propose a combination of character and word-level distances that can improve results, experiment with further methods, and report results.

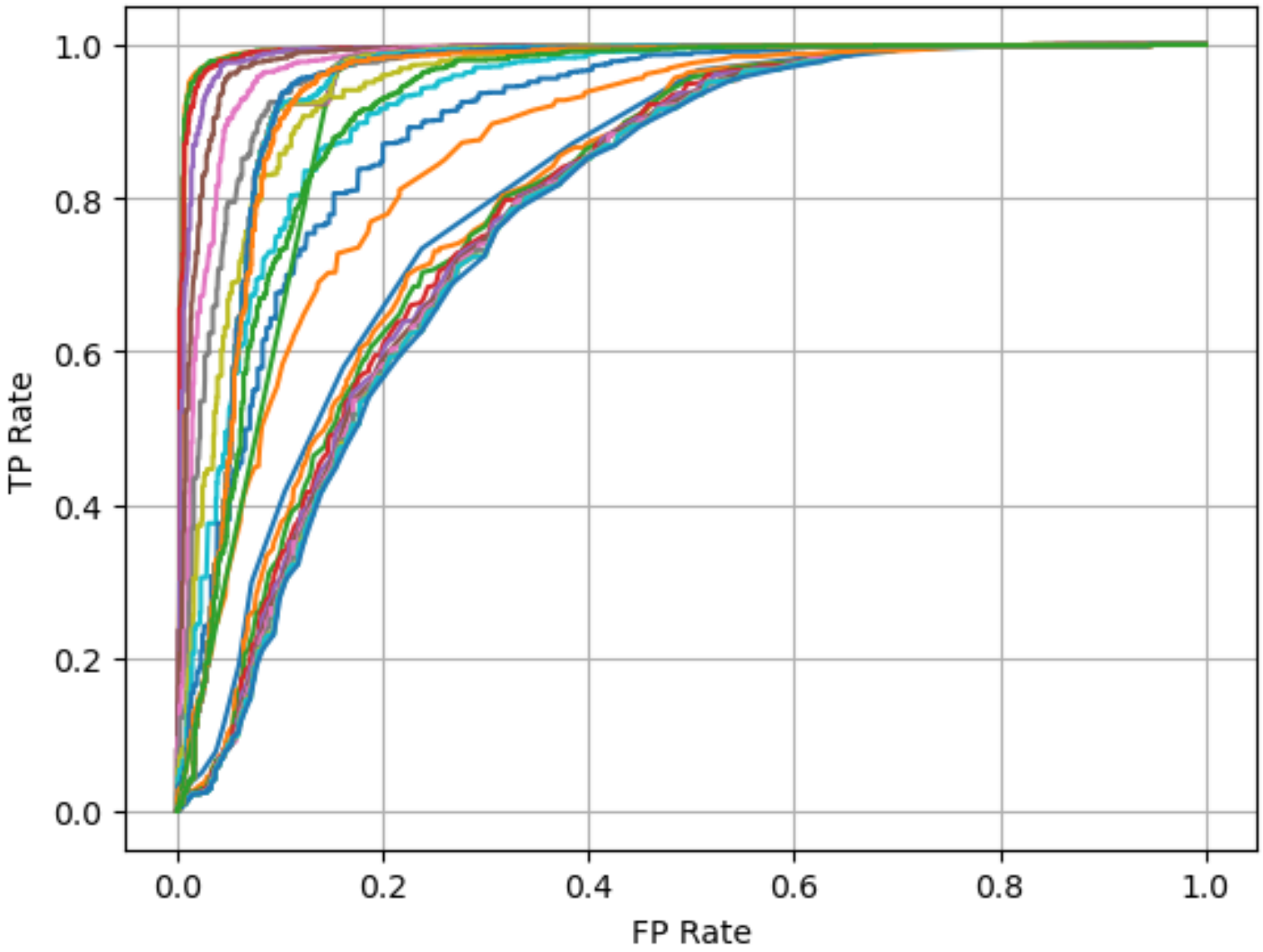

Fig. 5. All the methods with all possible combinations

The highest combination among all the methods will be equal to 0.993339645. Such a result was derived by combining the "Jaccard distance" method where "Jaccard distance char" has a weight equal to 0.1, and "Jaccard distance word" has a weight equal to 0.9.

7 Further work

- Keep researching and improving preprocessing methods. Use multiple stopword lists, experiment with different stemmers and lemmatizers, and assess the effects of various case sensitivity handling techniques, just to name a few variations. Examine the impact of these variations on the effectiveness of the distance metrics.
- Investigate handling methods for noise if data has it. When dealing with data that may contain typos or minor differences, need to think about fuzzy matching or approximation string matching techniques.
- Experiment with metrics other than ROC curves for evaluation. Depending on the particular issue you're seeking to resolve, take into account precision-recall curves, the F1-score, or other pertinent metrics.