Data Science Challenge

Methodology: Simulation framework

- The entries in `x_descriptions` of `train data` are converted into a set containing subsets of shingles. A MinHash is computed for set.
- The entries in `y_descriptions` of `train data` are treated in the same way.
- Similarity is computed between the MinHashs generated from `x_descriptions` and `y_descriptions` using Jaccard distance. The distance is a real number [0,1].
- The true labels for similarity are determined by `x_id` and `y_id`, where
 0: no-match & 1: match.
- ROC type simulation analysis is performed to determine the Jaccard threshold above which the Jaccard distances are deemed similar.
- The performance of the technique is examined using `test data`, using the jaccard threshold determined from the `train data` set

Methodology: Converting descriptions into sets

Word-level shingles

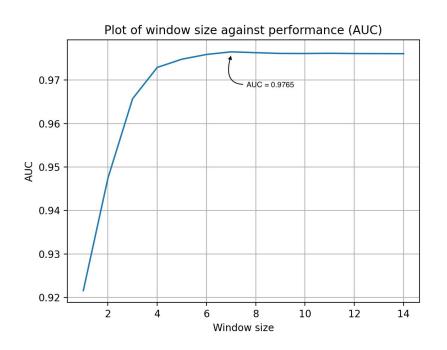
- Words are identified using spaces, everything is lower-case, punctuations are removed and a sliding window is applied to create subsets of words.
- Example:

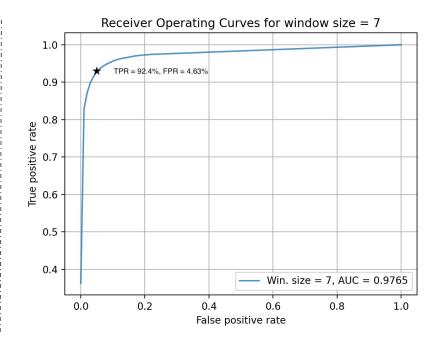
"If you can meet with Triumph and Disaster And treat those two impostors just the same"

```
Window size = 3: [[if, you, can], [you, can, meet], [can, meet, with], . . . , [impostors, just, the], [just, the, same]]
```

 We examine the performance across different window sizes, and select the window with the highest area under the curve (AUC).

Results: Word-level shingles





Results: Word-level shingles

- A window size = 7 yielded maximum auc of 0.9765 in train data and was applied on test data.
 The threshold was selected at TPR = 92.4% and FPR = 4.63%.
- The confusion matrix analyzes performance of the technique on test data.

Predicted	Actual	
	Match	No-match
Match	2431	118
No-match	123	1545

We observe 241 mis-classifications over 5762 data points

Results: Examining false negatives



- We examine the false negatives to understand scenarios that are not being identified as a match.
- We posit another technique of creating subsets from `descriptions`, based on characters (as opposed to words).

Methodology: Converting descriptions into sets

Character-level shingles

- Spaces are removed, everything is lower-case, punctuations are removed and a sliding window is applied to create subsets of characters.
- Example:

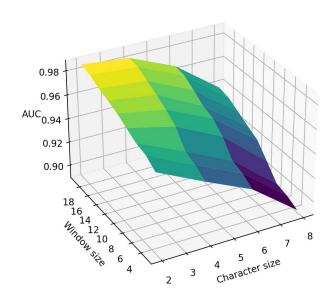
"If you can meet with Triumph and Disaster And treat those two impostors just the same"

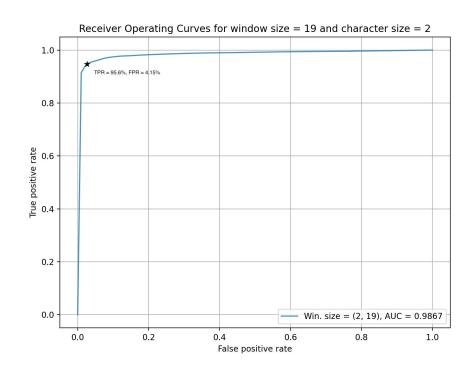
```
Character size = 2, Window size = 5: [[if, fy, yo, ou, uc], [fy, yo, ou, uc, ca], [yo, ou, uc, ca, an], . . . , [he,es,sa,am,me]]
```

 We examine the performance across different character sizes and window sizes, and select the combination with the highest area under the curve (AUC).

Results: Character-level shingles

AUC for combination of window and character sizes





Results: Character-level shingles

- A window size = 9 and character size = 2 yielded maximum auc of 0.9867 in train data and was applied on test data. The threshold was selected at TPR = 95.6% and FPR = 4.15%.
- The confusion matrix analyzes performance of the technique on test data.

Confusion matrix			
Predicted	Actual		
	Match	No-match	
Match	2434	115	
No-match	76	1592	

- We observe that character-level shingles has a better performance than word-level shingles, both in terms of reducing false negatives and false positives.
- A more rigorous search could be performed, for example across a more granular and across a larger span of character and window size combinations, to optimize for AUC.