Asmt 2: Document Similarity and Hashing

sushmithabhat09

January 2020

1 Creating k-Grams (50 points)

You will construct several types of k-grams for all documents. All documents only have at most 27 characters: all lower case letters and space. *Yes, the space counts as a character in character k-grams.*

[G1] Construct 2-grams based on characters, for all documents.

[G2] Construct 3-grams based on characters, for all documents.

[G3] Construct 2-grams based on words, for all documents.

Remember, that you should only store each k-gram once, duplicates are ignored.

A: (25 points) How many distinct k-grams are there for each document with each type of k-gram? You should report $4 \times 3 = 12$ different numbers.

	G1	G2	G3
D1	266	770	289
D2	264	759	297
D3	296	978	390
D4	249	770	364

B: (25 points) Compute the Jaccard similarity between all pairs of documents for each type of k-gram. You should report $3 \times 6 = 18$ different numbers.

	G1	G2	G3
JS(D1,D2)	0.99248	0.95524	0.79205
JS(D1,D3)	0.78413	0.50301	0.19542
JS(D1,D4)	0.66666	0.30619	0.00772
JS(D2,D3)	0.78344	0.49871	0.17637
JS(D2,D4)	0.66019	0.30349	0.00916
JS(D3,D4)	0.67178	0.313298	0.01208

2 Min Hashing (50 points)

We will consider a hash family H so that any hash function $h \in H$ maps from $h : \{\text{k}-grams\} \to [m]$ for m large enough (To be extra cautious, I suggest over $m \geq 10{,}000$; but should work with smaller m too).

A: (35 points) Using grams G2, build a min-hash signature for document `D1` and `D2` using $t = \{20, 60, 150, 300, 600\}$ hash functions. For each value of t report the approximate Jaccard similarity between the pair of documents `D1` and `D2`, estimating the Jaccard similarity:

$$\hat{}_t(a,b) = \frac{1}{t}\sum_{i=1}^{t} \{ 1 \text{ if } a_i = b_i 0 \text{if } a_i \neq b_i . You should report$$

$5 numbers.$

t	JS(D1,D2)
20	1.0
60	0.98333333
150	0.96
300	0.9366
600	0.95333

B: (15 point) What seems to be a good value for t? You may run more experiments. Justify your answer in terms of both accuracy and time.
Solution: Result of my experiments are captured in screenshot below:

t	JS	Time Taken(s)	Accuracy
20	1.0	0.04756784439086914	-0.04475703325
60	0.9833333333333333	0.13343405723571777	-0.02809036658141517
100	0.96	0.25095081329345703	-0.0047570332480818545
150	0.96	0.34028005599975586	-0.0047570332480818545
200	0.955	0.4554412364959717	0.00024296675191814998
250	0.964	0.5632390975952148	-0.008757033248081858
300	0.9366666666666666	0.6790487766265869	0.018576300085251463
350	0.9514285714285714	0.7898702621459961	0.0038143953233467087
400	0.965	0.9048440456390381	-0.009757033248081859
450	0.9733333333333334	1.0177040100097656	-0.018090366581415274
500	0.942	1.1259450912475586	0.013242966751918162
550	0.9472727272727273	1.239609956741333	0.007970239479190844
600	0.9533333333333334	1.3510830402374268	0.0019096334185847441
650	0.9507692307692308	1.493941068649292	0.004473735982687299
700	0.9671428571428572	1.7533900737762451	-0.011899890390939083
750	0.9533333333333334	1.8179740905761719	0.0019096334185847441
800	0.95	1.9492111206054688	0.005242966751918154
850	0.9635294117647059	1.9820408821105957	0.008286445012787746
900	0.9522222222222222	2.108820915222168	0.003020744529695918
950	0.9515789473684211	2.200716972351074	0.0036640193834970303
1000	0.957	2.352954864501953	-0.0017570332480818518

As we can see in the image, the accuracy increases as the value of t increases. But, time taken also increases as t increases. There is a tradeoff - compromise on accuracy then time taken will be less. I ran the experiment a couple of times and I can get some what good accuracy when t is between 150 to 300. In this particular experiment, the accuracy is the most when t = 200. For most of the runs, I was getting pretty good accuracy for t between 150 to 300. So the good value for t would be between 150 to 300. In this experiment t = 200.

3 Bonus (3 points)

Describe a scheme like Min-Hashing over a domain of size n for the *Andberg* Similarity, defined $(A, B) = \frac{|A \cap B|}{|A \cup B| + |A \triangle B|}$. That is so given two sets A and B and family of hash functions, then $\Pr_{h \in H}[h(A) = h(B)] = (A, B)$. Note the only randomness is in the choice of hash function h from the set H, and $h \in H$ represents the process of choosing a hash function (randomly) from H. The point of this question is to design this process, and show that it has the required property.

Or show that such a process cannot be done.