Text similarity detection by means of n-gram hashing

Maksym Sarana

m.sarana@ukma.edu.ua
National University of Kyiv-Mohyla Academy

Recent progress of computing technologies enable development of more sophisticated applications for natural language processing, in turn it requires improvement of algorithms and machine learning methods. One of the most common arising tasks are representation learning and comparison methods. Most of them are based on counting and cataloguing n-grams in streams of symbols. One of the fastest method of implementing such operations is by means of hash tables [1].

Given a sequence of symbols $S = \big(s_1, s_2, s_3, \ldots, s_{N+(n-1)}\big)$, an n-gram of the sequence is an n-long subsequence of consecutive symbols. The i-th n-gram of S is the sequence $(s_i, s_{i+1}, \ldots, s_{i+n-1})$. Note that there are N such n-grams in the sequence S.

The literature of hash functions and collision resolution schemes is extensive. A detailed analysis and an overview of early activity can be found in the classic work by Knuth [2].

The crucial point of the cross-validation method, that is widely used for machine learning models evaluation, is quality of the validation data set used, that must include previously unseen data only. An accidental data leak between the train and validation data sets may significantly corrupt results of the model evaluation. In this work we are considering application of hashed n-grams on the data leak search task for the textual data.

The dataset we use includes 4 groups of English text pairs, see Table 1.

Category	Ground truth	AVG #words	Number of pairs
slightly edited texts	same	785	100
random subsets (20-80%)	same	584	100
completely different texts	different	776	100
different parts of the same text	different	1146	100

Table 1: The groups of English text pairs in the dataset

Approach 1. The RAW 3-grams

First, we checked if it is possible to differentiate similar texts with sets of the unique 3-grams.

$$score = \frac{|G_1 \cap G_2|}{\min(|G_1|, |G_2|)},$$

where G_1 and G_2 — sets of unique 3-grams of the first and second texts respectively.

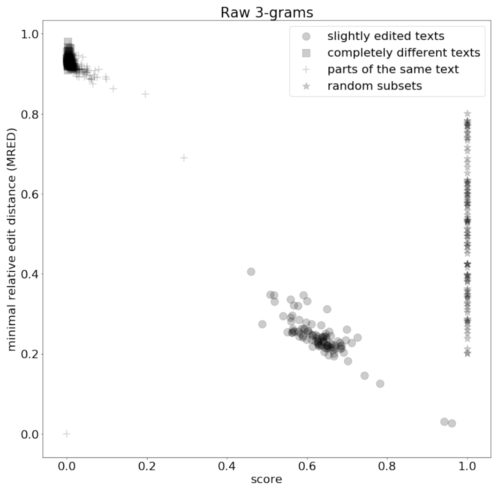

Figure 1: Scatter plot of score vs. MRED for raw trigrams

The scatter plot in Figure 1 compares the score with the minimal relative edit distance ($MRED$) between the texts:

$$MRED = \frac{Levenshtein(W_1, W_2)}{\min(|W_1|, |W_2|)},$$

where W_1 and W_2 are word sequences from the first and second texts respectively.

Approach 2. Use hashing

The one of widely used methods to compress arbitrary values to a fixed range of values is hashing. So, instead of the set of 3-grams we can operate with a set of their hashes. Since we are going to much narrower range of hashes than the Python hash function (SipHash) returns, we introduce hash size M as a hyperparameter of the algorithm, so that the 3-gram hash may be easily bounded by reminder of division on M:

$$h(g) = SipHash(g) \mod M,$$

where g is a 3-gram.

Consider an integer array of size M, where each element with index i includes number of unique hashed 3-grams with the hash i (i.e. number of hash collisions), but not more the maximal integer value for the given data type. The hash score in this case is

$$score = \frac{\sum_{i=1}^{M} \min(H_1^{(i)}, H_2^{(i)})}{\min(\sum_{j=1}^{M} H_1^{(j)}, \sum_{l=1}^{M} H_2^{(l)})},$$

where H_1 and H_2 are arrays with the number of collisions for the first and second texts respectively. The experiment has shown that perfect differentiation of the similar texts is possible if we use a *uint8* array of size $M = 2^{12}$, which takes 4096 bytes of memory.

Approach 3. Use boolean arrays

In this approach we just decreased the data type size of the array from *uint8* to *boolean*. So that, the i-th element of the array is *true* iff at least one 3-gram from the text has the hash value i. We can easily rewrite the *score* equation with the element wise *and* operation:

$$score = \frac{\sum_{i=1}^{M} [H_1^{(i)} \wedge H_2^{(i)}]}{\min(\sum_{j=1}^{M} [H_1^{(j)}], \sum_{l=1}^{M} [H_2^{(l)}])}.$$

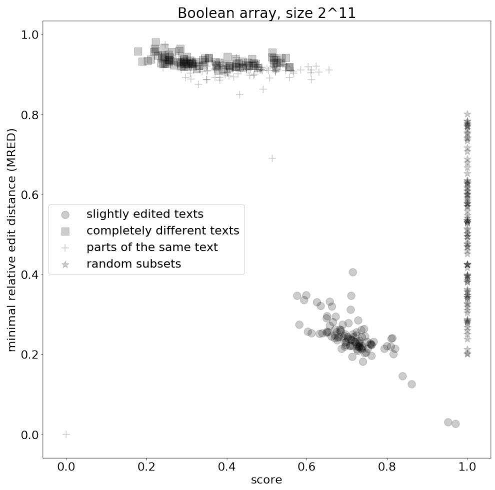

Figure 2: Scatter plot of score vs. MRED for hashed 3-grams and boolean array of size 2^{12}.

Results and conclusions

The results are present in the Table 2. The best threshold was chosen to maximize the F1-score.

Approach	Array type	Size M	Best threshold	F1-score	Memory
The raw 3-grams	-	-	0.459	1.0	unlimited
Hashing with a collision number	uint8	2^{11}	0.538	0.998	2048 bytes
	uint8	2^{12}	0.518	1.0	4096 bytes
Use a boolean array	boolean	2^{11}	0.575	0.980	256 bytes
	boolean	2^{12}	0.531	1.0	512 bytes

Table 2: Comparison of results of the approaches

We have found, that even 256-byte boolean array of size 2^{11} is able to store enough hashed 3-gram information for near-ideal similarity differentiation with F1-score 0.98.

Ground truth	Data category	Predicted	
		different	same
different	completely different texts	100	0
	parts of the same text	92	8
same	random subsets	0	100
	slightly edited texts	0	100

Table 3: Confusion matrix for text similarity differentiation

The confusion matrix is presented in Table 3. The few available errors are false positive, what is acceptable for the data leak detection task.

1. Jonathan D.C. Recursive hashing functions for n-grams // ACM Transactions on Information Systems. – 1997. – Vol. 15, Is. 3, – Pp. 291–320. https://doi.org/10.1145/256163.256168
2. Knuth D.E. The Art of Computer Programming. Vol. 3, Sorting and Searching. – Addison-Wesley, Reading, Mass. – 1973.