Sketching String Similarity

Yuchong Zhang

University of Toronto

December 8, 2021

Outline

Problem Background

My Algorithm for Weighted Jaccard Similarity Estimation

Some other interesting algorithms

Outline

Problem Background

My Algorithm for Weighted Jaccard Similarity Estimation

Some other interesting algorithms

Why do we care?

- Applications: search engine, DNA sequencing, intrusion detection, plagiarism detection etc.

Similarity Measurement

- $S \in \Sigma^*$: string. k-mers of S: substrings of length k.
  e.g.: AATT, TTCC are 4-mers of AATTCCGG.
  AATTCCGG   AATTCCGG

Similarity Measurement

- $S \in \Sigma^*$: string. k-mers of S: substrings of length k.
  e.g.: AATT, TTCC are 4-mers of AATTCCGG.
  AATTCCGG   AATTCCGG
- Given 2 sets A, B, the Jaccard Index $J(A,B) = \dfrac{|A \cap B|}{|A \cup B|}$.

Similarity Measurement

- $S \in \Sigma^*$: string. k-mers of S: substrings of length k.
  e.g.: AATT, TTCC are 4-mers of AATTCCGG.
  AATTCCGG AATTCCGG
- Given 2 sets A, B, the Jaccard Index $J(A,B) = \dfrac{|A \cap B|}{|A \cup B|}$.
- $J_k(S_1, S_2) :=$ Jaccard Index of k-mer sets.
- Recall: MinHash Sketching estimate $J_k(S_1, S_2)$.

Limitation to Jaccard Similarity

By treating S_1, S_2 as sets of k-mers and use $J_k(S_1, S_2)$, we ignore:

- k-mer frequency.
- order in which k-mers appear.

Both factors can affect string similarity.

Limitation to Jaccard Similarity

e.g.: consider:
$S_1 = 000000000000\textcolor{red}{1}00000000000000000000$
$S_2 = 000000000000000000000000000000000$

- Intuitively: S_1, S_2 should be pertty similar (1 character difference.)

Limitation to Jaccard Similarity

e.g.: consider:
$S_1 = 000000000000{\color{red}1}00000000000000000000$
$S_2 = 000000000000000000000000000000000$

- Intuitively: S_1, S_2 should be pertty similar (1 character difference.)
- But $J_k(S_1, S_2) = \dfrac{1}{k+1} \leq 0.5$.

Limitation to Jaccard Similarity

$S_1 = 000000000000{\color{red}1}00000000000000000000$
$S_2 = 000000000000000000000000000000000$

Idea: even though the "1" in S_1 created k extra k-mers, they should not affect the similarity too much because each of them only appeared once.

Account for frequency!

Limitation of Jaccard Similarity

Might want better measurement of similarity.

- Classical: editing distance.
- We will use a simpler method that extends $J_k(S_1, S_2)$.

Weighted Jaccard Similarity

- For k-mer σ and string S, define $N_k(\sigma, S) :=$ number of times σ appears in S (0 if not present).

Weighted Jaccard Similarity

- For k-mer σ and string S, define $N_k(\sigma, S) :=$ number of times σ appears in S (0 if not present).
- The Weighted Jaccard Similarity between strings S_1, S_2 is defined as

$$J_k^w(S_1, S_2) := \frac{\sum_\sigma \min\{N_k(\sigma, S_1), N_k(\sigma, S_2)\}}{\sum_\sigma \max\{N_k(\sigma, S_1), N_k(\sigma, S_2)\}}$$

Weighted Jaccard Similarity

- For k-mer σ and string S, define $N_k(\sigma, S) :=$ number of times σ appears in S (0 if not present).
- The Weighted Jaccard Similarity between strings S_1, S_2 is defined as

$$J_k^w(S_1, S_2) := \frac{\sum_\sigma \min\{N_k(\sigma, S_1), N_k(\sigma, S_2)\}}{\sum_\sigma \max\{N_k(\sigma, S_1), N_k(\sigma, S_2)\}}$$

- If ATCG appears 10 times in S_1 and 4 times in S_2, treat the first 4 occurences as "the same".

Weighted Jaccard Similarity

$$J_k^w(S_1, S_2) := \frac{\sum_\sigma \min\{N_k(\sigma, S_1), N_k(\sigma, S_2)\}}{\sum_\sigma \max\{N_k(\sigma, S_1), N_k(\sigma, S_2)\}}$$

Weighted Jaccard Similarity

$$J_k^w(S_1,S_2) := \frac{\sum_\sigma \min\{N_k(\sigma,S_1),N_k(\sigma,S_2)\}}{\sum_\sigma \max\{N_k(\sigma,S_1),N_k(\sigma,S_2)\}}$$

e.g.:
$S_1 =$AAAAAAT
$S_2 =$AAAAAT

The 4-mers are $\sigma =$ AAAA and $\tau =$ AAAT.

$$N_4(\sigma,S_1)=3, N_4(\sigma,S_2)=2, N_4(\tau,S_1)=N_4(\tau,S_2)=1$$

So

$$J_4^w(S_1,S_2)=\frac{2+1}{3+1}=\frac{3}{4}=0.75$$

In comparison, $J_4(S_1,S_2)=1$. So the Weighted Jaccard Similarity picks up the difference that S_1 has 1 additional A compared to S_2.

Weighted Jaccard Similarity

$J_k^w(S_1, S_2)$ extends $J_k(S_1, S_2)$?

- Append to each k-mer σ at certain position its occurrence number i: $i =$ the number of times σ appears in S so far.

Weighted Jaccard Similarity

$J_k^w(S_1, S_2)$ extends $J_k(S_1, S_2)$?

- Append to each k-mer σ at certain position its occurrence number i: $i =$ the number of times σ appears in S so far.
- (σ, i): an indexed k-mer of σ.

e.g.: in string ATCGCCCCATCGTTTTATCG,
(ATCG, 1), (ATCG, 2), (ATCG, 3) are all the indexed 4-mers of ATCG

Weighted Jaccard Similarity

$J_k^w(S_1, S_2)$ extends $J_k(S_1, S_2)$?

- Append to each k-mer σ at certain position its occurrence number i: $i =$ the number of times σ appears in S so far.
- (σ, i): an indexed k-mer of σ.

e.g.: in string ATCGCCCCATCGTTTTATCG,
(ATCG, 1), (ATCG, 2), (ATCG, 3) are all the indexed 4-mers of ATCG

If we let $T_k(S)$ denote the set of indexed k-mers of S, then

$$J_k^w(S_1, S_2) = J(T_k(S_1), T_k(S_2)) = \frac{|T_k(S_1) \cap T_k(S_2)|}{|T_k(S_1) \cup T_k(S_2)|}$$

Weighted Jaccard Similarity

$$J_k^w(S_1, S_2) = J(T_k(S_1), T_k(S_2)) = \frac{|T_k(S_1) \cap T_k(S_2)|}{|T_k(S_1) \cup T_k(S_2)|}$$

e.g.: $S_1 =$AAAAAA
$S_2 =$AAAA

$T_4(S_1) = \{$(AAAA, 1), (AAAA, 2), (AAAA, 3)$\}$
$T_4(S_2) = \{$(AAAA,1)$\}$.

Weighted Jaccard Similarity

$$J_k^w(S_1, S_2) = J(T_k(S_1), T_k(S_2)) = \frac{|T_k(S_1) \cap T_k(S_2)|}{|T_k(S_1) \cup T_k(S_2)|}$$

e.g.: $S_1 =$AAAAAA
$S_2 =$AAAA

$T_4(S_1) = \{$(AAAA, 1), (AAAA, 2), (AAAA, 3)$\}$
$T_4(S_2) = \{$(AAAA,1)$\}$.

$$J_4^w(S_1, S_2) = \frac{1}{3} = \frac{|\{(AAAA, 1)\}|}{|\{(AAAA, 1), (AAAA, 2), (AAAA, 3)\}|}.$$

Compute Weighted Jaccard Similarity

Although $J_k^w(S_1, S_2) = J(T_k(S_1), T_k(S_2))$:

- can't simply run MinHash to compute J_k^w if we don't know $T_k(S_1), T_k(S_2)$.
- Brute force: store all their k-mer frequencies and compute $J_k^w(S_1, S_2)$ exactly.
- Low memory?

Outline

Problem Background

My Algorithm for Weighted Jaccard Similarity Estimation

Some other interesting algorithms

Intuition

MinHash: construct an event (collision of minimum hash values in S_1, S_2) whose probability is $J_k(S_1, S_2)$, the quantity MinHash tries to estimate.

Intuition

MinHash: construct an event (collision of minimum hash values in S_1, S_2) whose probability is $J_k(S_1, S_2)$, the quantity MinHash tries to estimate.

I wanted to follow the same idea. The goal is to estimate

$$J_k^w(S_1, S_2) = \frac{\sum_\sigma \min\{N_k(\sigma, S_1), N_k(\sigma, S_2)\}}{\sum_\sigma \max\{N_k(\sigma, S_1), N_k(\sigma, S_2)\}} = \frac{|T_k(S_1) \cap T_k(S_2)|}{|T_k(S_1) \cup T_k(S_2)|}$$

Want: event with prob $\frac{|T_k(S_1) \cap T_k(S_2)|}{|T_k(S_1) \cup T_k(S_2)|}$.

Intuition

Want: event with prob $\dfrac{|T_k(S_1) \cap T_k(S_2)|}{|T_k(S_1) \cup T_k(S_2)|}$.

Intuition

Want: event with prob $\dfrac{|T_k(S_1) \cap T_k(S_2)|}{|T_k(S_1) \cup T_k(S_2)|}$.

Candidate: random sample from $T_k(S_1) \cup T_k(S_2)$ falls in $T_k(S_1) \cap T_k(S_2)$.

Intuition

Want: event with prob $\frac{|T_k(S_1) \cap T_k(S_2)|}{|T_k(S_1) \cup T_k(S_2)|}$.

Candidate: random sample from $T_k(S_1) \cup T_k(S_2)$ falls in $T_k(S_1) \cap T_k(S_2)$.

Sample (σ, i) is in $T_k(S_1) \cap T_k(S_2) \iff i \leq \min\{N_k(\sigma, S_1), N_k(\sigma, S_2)\}$, the smaller frequency.

Intuition

Problem: don't know $T_k(S_1), T_k(S_2)$ a priori

Intuition

Problem: don't know $T_k(S_1), T_k(S_2)$ a priori

We need:
$S_1 = \ldots(\sigma, 1)\ldots(\sigma, 2)\ldots(\sigma, l)\ldots(\sigma, m)\ldots$
$S_2 = \ldots(\sigma, 1)\ldots(\sigma, 2)\ldots(\sigma, l)\ldots$

Intuition

Problem: don't know $T_k(S_1), T_k(S_2)$ a priori

We need:
$S_1 = \ldots(\sigma, 1)\ldots(\sigma, 2)\ldots(\sigma, l)\ldots(\sigma, m)\ldots$
$S_2 = \ldots(\sigma, 1)\ldots(\sigma, 2)\ldots(\sigma, l)\ldots$
i.e., need to "match" first l copies and treat them as same in sample space.

Intuition

Problem: don't know $T_k(S_1), T_k(S_2)$ a priori

We need:
$S_1 = \ldots(\sigma,1)\ldots(\sigma,2)\ldots(\sigma,l)\ldots(\sigma,m)\ldots$
$S_2 = \ldots(\sigma,1)\ldots(\sigma,2)\ldots(\sigma,l)\ldots$
i.e., need to "match" first l copies and treat them as same in sample space.

Reality:
$S_1 = \ldots\sigma\ldots\sigma\ldots\sigma\ldots\sigma\ldots$
$S_2 = \ldots\sigma\ldots\sigma\ldots\sigma\ldots$

Intuition

Problem: don't know $T_k(S_1), T_k(S_2)$ a priori

We need:
$S_1 = \ldots (\sigma, 1) \ldots (\sigma, 2) \ldots (\sigma, l) \ldots (\sigma, m) \ldots$
$S_2 = \ldots (\sigma, 1) \ldots (\sigma, 2) \ldots (\sigma, l) \ldots$
i.e., need to "match" first l copies and treat them as same in sample space.

Reality:
$S_1 = \ldots \sigma \ldots \sigma \ldots \sigma \ldots \sigma \ldots$
$S_2 = \ldots \sigma \ldots \sigma \ldots \sigma \ldots$

All σ's are made distinct by their positions in strings.

Intuition

Reality:
$S_1 = \ldots\sigma\ldots\sigma\ldots\sigma\ldots\sigma\ldots$
$S_2 = \ldots\sigma\ldots\sigma\ldots\sigma\ldots$

All σ's are made distinct by their positions in strings.

Available sample space: all copies of k-mers treated distinct. size = $\sum_{\tau} N_k(\tau, S_1) + N_k(\tau, S_2)$ vs

hope: $T_k(S_1) \cup T_k(S_2)$, size $= \sum_{\tau} \max\{N_k(\tau, S_1), N_k(\tau, S_2)\}$.

Intuition

Reality:
$S_1 = \dots \sigma \dots \sigma \dots \sigma \dots \sigma \dots$
$S_2 = \dots \sigma \dots \sigma \dots \sigma \dots$

All σ's are made distinct by their positions in strings.

Available sample space: all copies of k-mers treated distinct. size = $\sum_{\tau} N_k(\tau, S_1) + N_k(\tau, S_2)$ vs
hope: $T_k(S_1) \cup T_k(S_2)$, size $= \sum_{\tau} \max\{N_k(\tau, S_1), N_k(\tau, S_2)\}$.

But:

- if we know position of a σ, then can compute its occurrence number m in string. So can still check if $(\sigma, m) \in T_k(S_1) \cap T_k(S_2)$.

Intuition

Reality:
$S_1 = \ldots\sigma\ldots\sigma\ldots\sigma\ldots\sigma\ldots$
$S_2 = \ldots\sigma\ldots\sigma\ldots\sigma\ldots$

All σ's are made distinct by their positions in strings.

Available sample space: all copies of k-mers treated distinct. size = $\sum_{\tau} N_k(\tau, S_1) + N_k(\tau, S_2)$ vs

hope: $T_k(S_1) \cup T_k(S_2)$, size $= \sum_{\tau} \max\{N_k(\tau, S_1), N_k(\tau, S_2)\}$.

But:

- if we know position of a σ, then can compute its occurrence number m in string. So can still check if $(\sigma, m) \in T_k(S_1) \cap T_k(S_2)$.
- Know exactly how much larger available sample space vs hope.

Intuition

Reality:
$S_1 = \dots\sigma\dots\sigma\dots\sigma\dots\sigma\dots$
$S_2 = \dots\sigma\dots\sigma\dots\sigma\dots$

All σ's are made distinct by their positions in strings.

Available sample space: all copies of k-mers treated distinct. size = $\sum_\tau N_k(\tau, S_1) + N_k(\tau, S_2)$ vs

hope: $T_k(S_1) \cup T_k(S_2)$, size $= \sum_\tau \max\{N_k(\tau, S_1), N_k(\tau, S_2)\}$.

But:

- if we know position of a σ, then can compute its occurrence number m in string. So can still check if $(\sigma, m) \in T_k(S_1) \cap T_k(S_2)$.
- Know exactly how much larger available sample space vs hope.
- Maybe: can sample from larger space. Compensate later.

- Treat all k-mer copies as distinct by positions.
- Sample k-mer σ at a random position anyway.

- Treat all k-mer copies as distinct by positions.
- Sample k-mer σ at a random position anyway.
- Compute the occurrence number m

- Treat all k-mer copies as distinct by positions.
- Sample k-mer σ at a random position anyway.
- Compute the occurrence number m
- What is $\mathbb{P}[(\sigma, m)$ is in $T_k(S_1) \cap T_k(S_2)]$?

- Treat all k-mer copies as distinct by positions.
- Sample k-mer σ at a random position anyway.
- Compute the occurrence number m
- What is $\mathbb{P}[(\sigma, m)$ is in $T_k(S_1) \cap T_k(S_2)]$?

Does $\mathbb{P}$ = $J_k^w(S_1, S_2) = \dfrac{|T_k(S_1) \cap T_k(S_2)|}{|T_k(S_1) \cup T_k(S_2)|}$?

- Treat all k-mer copies as distinct by positions.
- Sample k-mer σ at a random position anyway.
- Compute the occurrence number m
- What is $\mathbb{P}[(\sigma, m)$ is in $T_k(S_1) \cap T_k(S_2)]$?

Does $\mathbb{P}$ = $J_k^w(S_1, S_2) = \dfrac{|T_k(S_1) \cap T_k(S_2)|}{|T_k(S_1) \cup T_k(S_2)|}$?

- Probably not... True sample space is not $T_k(S_1) \cup T_k(S_2)$.

- Treat all k-mer copies as distinct by positions.
- Sample k-mer σ at a random position anyway.
- Compute the occurrence number m
- What is $\mathbb{P}[(\sigma, m)$ is in $T_k(S_1) \cap T_k(S_2)]$?

Does $\mathbb{P}$ = $J_k^w(S_1, S_2) = \dfrac{|T_k(S_1) \cap T_k(S_2)|}{|T_k(S_1) \cup T_k(S_2)|}$?

- Probably not... True sample space is not $T_k(S_1) \cup T_k(S_2)$.
- But we know how much larger True sample space is.
- Can we use this to transform $\mathbb{P}$ to $J_k^w(S_1, S_2)$?

- Treat all k-mer copies as distinct by positions.
- Sample k-mer σ at a random position anyway.
- Compute the occurrence number m
- What is $\mathbb{P}[(\sigma, m)$ is in $T_k(S_1) \cap T_k(S_2)]$?

Does $\mathbb{P}$ = $J_k^w(S_1, S_2) = \dfrac{|T_k(S_1) \cap T_k(S_2)|}{|T_k(S_1) \cup T_k(S_2)|}$?

- Probably not... True sample space is not $T_k(S_1) \cup T_k(S_2)$.
- But we know how much larger True sample space is.
- Can we use this to transform $\mathbb{P}$ to $J_k^w(S_1, S_2)$?

Turns out, $\mathbb{P}$ can be transformed into $J_k^w(S_1, S_2)$!

My Algorithm

$f(S_1, S_2, k)$. Inputs: strings S_1, S_2 and integer k.

My Algorithm

$f(S_1, S_2, k)$. Inputs: strings S_1, S_2 and integer k.

1. Sample a position i uniformly at random from all possible positions in S_1, S_2 where a k-mer resides.

My Algorithm

$f(S_1, S_2, k)$. Inputs: strings S_1, S_2 and integer k.

1. Sample a position i uniformly at random from all possible positions in S_1, S_2 where a k-mer resides.
2. Let $S :=$ the string containing position i, $S' :=$ the other string. $\sigma :=$ the k-mer at position i in S. (σ and i is our sketch.)

My Algorithm

$f(S_1, S_2, k)$. Inputs: strings S_1, S_2 and integer k.

1. Sample a position i uniformly at random from all possible positions in S_1, S_2 where a k-mer resides.
2. Let $S :=$ the string containing position i, $S' :=$ the other string. $\sigma :=$ the k-mer at position i in S. (σ and i is our sketch.)
3. Pass through S, find $m \leftarrow$ number of occurrences of σ in S up to and including position i. (m = Occurrence number)

My Algorithm

$f(S_1, S_2, k)$. Inputs: strings S_1, S_2 and integer k.

1. Sample a position i uniformly at random from all possible positions in S_1, S_2 where a k-mer resides.
2. Let $S :=$ the string containing position i, $S' :=$ the other string. $\sigma :=$ the k-mer at position i in S. (σ and i is our sketch.)
3. Pass through S, find $m \leftarrow$ number of occurrences of σ in S up to and including position i. (m = Occurrence number)

   Next: Check if $(\sigma, m) \in T_k(S) \cap T_k(S')$
4. Pass through S', check if σ appears at least m times in S'. Return 1 if true, 0 otherwise.

My Algorithm

Example:

$k = 4$
$S_1 =$ AATTCCGGAATTCCGG
$S_2 =$ AATTCCGG

My Algorithm

Example:

$k = 4$
$S_1 =$ AATTCCGGAATTCCGG
$S_2 =$ AATTCCGG

Suppose random sample $i =$ position 1 in S_1:

A A T T C C G G A A T T C C G G
0 1 2 3 . . .

My Algorithm

Example:

$k = 4$
$S_1 =$ AATTCCGGAATTCCGG
$S_2 =$ AATTCCGG

Suppose random sample $i =$ position 1 in S_1:

A A T T C C G G A A T T C C G G
0 1 2 3 . . .

sample: (ATTC, 1) in $T_4(S_1)$.

My Algorithm

Example:

$k = 4$
$S_1 =$ AATTCCGGAATTCCGG
$S_2 =$ AATTCCGG

Suppose random sample $i =$ position 1 in S_1:

A A T T C C G G A A T T C C G G
0 1 2 3 . . .

sample: (ATTC, 1) in $T_4(S_1)$.

Clearly: (ATTC, 1) is also in $T_4(S_2)$. So $f(S_1, S_2, 4)$ returns 1.

f Explained

- $f(S_1, S_2, k)$ is the indicator of the event that the random sample $(\sigma, m) \in T_k(S_1) \cap T_k(S_2)$.

f Explained

- $f(S_1, S_2, k)$ is the indicator of the event that the random sample $(\sigma, m) \in T_k(S_1) \cap T_k(S_2)$.
- Because we didn't exactly sample (σ, m) from $T_k(S_1) \cup T_k(S_2)$, $p = \mathbb{P}[f = 1]$ probably won't be $J_k^w(S_1, S_2)$.

f Explained

- $f(S_1, S_2, k)$ is the indicator of the event that the random sample $(\sigma, m) \in T_k(S_1) \cap T_k(S_2)$.
- Because we didn't exactly sample (σ, m) from $T_k(S_1) \cup T_k(S_2)$, $p = \mathbb{P}[f = 1]$ probably won't be $J_k^w(S_1, S_2)$.
- But we hope we can transform p into $J_k^w(S_1, S_2)$ somehow.

f Explained

- $f(S_1, S_2, k)$ is the indicator of the event that the random sample $(\sigma, m) \in T_k(S_1) \cap T_k(S_2)$.
- Because we didn't exactly sample (σ, m) from $T_k(S_1) \cup T_k(S_2)$, $p = \mathbb{P}[f = 1]$ probably won't be $J_k^w(S_1, S_2)$.
- But we hope we can transform p into $J_k^w(S_1, S_2)$ somehow.

Analysis of p:

f Explained

- $f(S_1, S_2, k)$ is the indicator of the event that the random sample $(\sigma, m) \in T_k(S_1) \cap T_k(S_2)$.
- Because we didn't exactly sample (σ, m) from $T_k(S_1) \cup T_k(S_2)$, $p = \mathbb{P}[f = 1]$ probably won't be $J_k^w(S_1, S_2)$.
- But we hope we can transform p into $J_k^w(S_1, S_2)$ somehow.

Analysis of p:

Let σ be any k-mer and suppose S_1, S_2 look like:

$$S_1 = \sigma \ldots \sigma \ldots \sigma \ldots \sigma \ldots \sigma \ldots$$
$$S_2 = \sigma \ldots \sigma \ldots \sigma \ldots \sigma \ldots$$

where σ and σ mark the last copy of σ in S_1, S_2, respectively. Then by definition, σ is the $N_k(\sigma, S_2)$th copy of σ in S_2. Let σ be the $N_k(\sigma, S_2)$th copy of σ in S_1.

f Explained

Sample = σ and f returns 1 happens in this region:

$$\begin{aligned} S_1 &= \sigma \dots \sigma \dots \sigma \dots \sigma \dots \sigma \dots \\ S_2 &= \sigma \dots \sigma \dots \sigma \dots \sigma \dots \end{aligned}$$

f Explained

Sample = σ and f returns 1 happens in this region:

$$\begin{aligned} S_1 &= \boxed{\sigma \ldots \sigma \ldots \sigma \ldots \sigma} \ldots \sigma \ldots \\ S_2 &= \boxed{\sigma \ldots \sigma \ldots \sigma \ldots \sigma} \ldots \end{aligned}$$

The circled region above contains
$2 \cdot N_k(\sigma, S_2) = 2 \min\{N_k(\sigma, S_1), N_k(\sigma, S_2)\}$ copies of σ.

f Explained

Sample = σ and f returns 1 happens in this region:

$$\begin{aligned} S_1 &= \sigma \ldots \sigma \ldots \sigma \ldots \sigma \ldots \sigma \ldots \\ S_2 &= \sigma \ldots \sigma \ldots \sigma \ldots \sigma \ldots \end{aligned}$$

The circled region above contains
$2 \cdot N_k(\sigma, S_2) = 2\min\{N_k(\sigma, S_1), N_k(\sigma, S_2)\}$ copies of σ.

There are in total $\sum_\tau N_k(\tau, S_1) + N_k(\tau, S_2)$ elements in the sample space (the total number of k-mer copies).

f Explained

Sample = σ and f returns 1 happens in this region:

$$\begin{aligned} S_1 &= \sigma \ldots \sigma \ldots \sigma \ldots \sigma \ldots \sigma \ldots \\ S_2 &= \sigma \ldots \sigma \ldots \sigma \ldots \sigma \ldots \end{aligned}$$

The circled region above contains
$2 \cdot N_k(\sigma, S_2) = 2 \min\{N_k(\sigma, S_1), N_k(\sigma, S_2)\}$ copies of σ.

There are in total $\sum\limits_{\tau} N_k(\tau, S_1) + N_k(\tau, S_2)$ elements in the sample space (the total number of k-mer copies).

Because of uniform sampling:

$$\mathbb{P}[f \text{ returns 1 and sample is } \sigma] = \frac{2 \min\{N_k(\sigma, S_1), N_k(\sigma, S_2)\}}{\sum_\tau N_k(\tau, S_1) + N_k(\tau, S_2)}$$

f Explained

$$\mathbb{P}[f \text{ returns 1 and sample is } \sigma] = \frac{2\min\{N_k(\sigma, S_1), N_k(\sigma, S_2)\}}{\sum_\tau N_k(\tau, S_1) + N_k(\tau, S_2)}$$

f Explained

$$\mathbb{P}[f \text{ returns 1 and sample is } \sigma] = \frac{2\min\{N_k(\sigma, S_1), N_k(\sigma, S_2)\}}{\sum_\tau N_k(\tau, S_1) + N_k(\tau, S_2)}$$

Thus:

$$p = \mathbb{P}[f \text{ returns 1}] = \frac{2\sum_\tau \min\{N_k(\tau, S_1), N_k(\tau, S_2)\}}{\sum_\tau N_k(\tau, S_1) + N_k(\tau, S_2)}$$

f Explained

$$\mathbb{P}[f \text{ returns 1 and sample is } \sigma] = \frac{2\min\{N_k(\sigma, S_1), N_k(\sigma, S_2)\}}{\sum_\tau N_k(\tau, S_1) + N_k(\tau, S_2)}$$

Thus:

$$p = \mathbb{P}[f \text{ returns 1}] = \frac{2\sum_\tau \min\{N_k(\tau, S_1), N_k(\tau, S_2)\}}{\sum_\tau N_k(\tau, S_1) + N_k(\tau, S_2)}$$

This is not quite what we want:

$$J_k^w(S_1, S_2) = \frac{\sum_\tau \min\{N_k(\tau, S_1), N_k(\tau, S_2)\}}{\sum_\tau \max\{N_k(\tau, S_1), N_k(\tau, S_2)\}}$$

f Explained

But, we can write:

$$\begin{aligned} p &= \frac{2\sum_\tau \min\{N_k(\tau, S_1), N_k(\tau, S_2)\}}{\sum_\tau N_k(\tau, S_1) + N_k(\tau, S_2)} \\ &= \frac{2\sum_\tau \min\{N_k(\tau, S_1), N_k(\tau, S_2)\}}{\sum_\tau \min\{N_k(\tau, S_1), N_k(\tau, S_2)\} + \max\{N_k(\tau, S_1), N_k(\tau, S_2)\}} \end{aligned}$$

f Explained

But, we can write:

$$p = \frac{2\sum_\tau \min\{N_k(\tau, S_1), N_k(\tau, S_2)\}}{\sum_\tau N_k(\tau, S_1) + N_k(\tau, S_2)}$$
$$= \frac{2\sum_\tau \min\{N_k(\tau, S_1), N_k(\tau, S_2)\}}{\sum_\tau \min\{N_k(\tau, S_1), N_k(\tau, S_2)\} + \max\{N_k(\tau, S_1), N_k(\tau, S_2)\}}$$

and solve:

$$\frac{p}{2-p} = \frac{\sum_\tau \min\{N_k(\tau, S_1), N_k(\tau, S_2)\}}{\sum_\tau \max\{N_k(\tau, S_1), N_k(\tau, S_2)\}}$$
$$= J_k^w(S_1, S_2)$$

My Algorithm

To estimate $J_k^w(S_1, S_2, k)$, simply run $g(S_1, S_2, k)$:

1. Run R independent copies of $f(S_1, S_2, k)$. $\delta_j =$ return value of the jth copy.
2. $\hat{p} = \frac{1}{R}\sum_{j=1}^{R} \delta_j$.
3. Output $\frac{\hat{p}}{2-\hat{p}}$ as the estimate of $J_k^w(S_1, S_2)$.

In step 1 and 2, $\hat{p} = \frac{1}{R}\sum_{j=1}^{R} \delta_j$ is simply an estimate of the true probability $p = \mathbb{P}[f$ returns 1$]$.

Accuracy Analysis

Because:

- $\mathbb{E}[\hat{p}] = \frac{1}{R}\sum_{j=1}^{R}\mathbb{E}[\delta_j] = p$
- $J_k^w(S_1, S_2) = \frac{p}{2-p}$

Accuracy Analysis

Because:

- $\mathbb{E}[\hat{p}] = \frac{1}{R}\sum_{j=1}^{R}\mathbb{E}[\delta_j] = p$
- $J_k^w(S_1, S_2) = \frac{p}{2-p}$

we can hope that $\frac{\hat{p}}{2-\hat{p}}$ will not be too far from $J_k^w(S_1, S_2)$ if $\hat{p}$ is not too far from p.

Accuracy Analysis

Indeed:

$$|\frac{p}{2-p} - \frac{\hat{p}}{2-\hat{p}}| = \frac{2|p-\hat{p}|}{(2-p)(2-\hat{p})} \leq 2|p-\hat{p}|$$

Accuracy Analysis

Indeed:

$$\left|\frac{p}{2-p} - \frac{\hat{p}}{2-\hat{p}}\right| = \frac{2|p-\hat{p}|}{(2-p)(2-\hat{p})} \leq 2|p-\hat{p}|$$

So if $\hat{p}$ is within ϵ of p, then $\frac{\hat{p}}{2-\hat{p}}$ will be within 2ϵ of $J_k^w(S_1, S_2)$.

Accuracy Analysis

Indeed:

$$|\frac{p}{2-p} - \frac{\hat{p}}{2-\hat{p}}| = \frac{2|p-\hat{p}|}{(2-p)(2-\hat{p})} \leq 2|p-\hat{p}|$$

So if $\hat{p}$ is within ϵ of p, then $\frac{\hat{p}}{2-\hat{p}}$ will be within 2ϵ of $J_k^w(S_1, S_2)$.

Can apply Chebyshev's inequality to obtain:

$$\mathbb{P}[|\hat{p} - p| \geq \epsilon] \leq \frac{p - p^2}{R\epsilon^2}$$

Accuracy Analysis

Indeed:

$$|\frac{p}{2-p} - \frac{\hat{p}}{2-\hat{p}}| = \frac{2|p-\hat{p}|}{(2-p)(2-\hat{p})} \leq 2|p-\hat{p}|$$

So if $\hat{p}$ is within ϵ of p, then $\frac{\hat{p}}{2-\hat{p}}$ will be within 2ϵ of $J_k^w(S_1, S_2)$.

Can apply Chebyshev's inequality to obtain:

$$\mathbb{P}[|\hat{p}-p| \geq \epsilon] \leq \frac{p-p^2}{R\epsilon^2}$$

Simulation: if f is repeated 100 times, then with 0.9 probability, $\frac{\hat{p}}{2-\hat{p}}$ will be within 5 percent of $J_k^w(S_1, S_2)$.

Complexity analysis

- runtime(g) = $O(R \cdot \text{runtime}(f))$ (run f R times)

Complexity analysis

- runtime(g) = $O(R \cdot \text{runtime}(f))$ (run f R times)
- runtime(f) = $O(\text{time for sampling} + |S_1| + |S_2|)$
  (only need 1 pass after sampling)

Complexity analysis

- runtime(g) = $O(R \cdot \text{runtime}(f))$ (run f R times)
- runtime(f) = $O(\text{time for sampling} + |S_1| + |S_2|)$
  (only need 1 pass after sampling)
- Sample a random position: random integer in certain range
  (need to know $|S_1|, |S_2|$) $\implies$ $O(1)$ time and space.

Complexity analysis

- runtime(g) = $O(R \cdot \text{runtime}(f))$ (run f R times)
- runtime(f) = $O(\text{time for sampling} + |S_1| + |S_2|)$ (only need 1 pass after sampling)
- Sample a random position: random integer in certain range (need to know $|S_1|, |S_2|$) $\implies$ $O(1)$ time and space.
- Thus: runtime(f) = $O(|S_1| + |S_2|)$, runtime$(g) = O(R(|S_1| + |S_2|))$.

Complexity analysis

- runtime(g) = $O(R \cdot \text{runtime}(f))$ (run f R times)
- runtime(f) = $O(\text{time for sampling} + |S_1| + |S_2|)$ (only need 1 pass after sampling)
- Sample a random position: random integer in certain range (need to know $|S_1|, |S_2|$) $\implies$ $O(1)$ time and space.
- Thus: runtime(f) = $O(|S_1| + |S_2|)$, runtime$(g) = O(R(|S_1| + |S_2|))$.
- f only need $O(k)$ space to store k-mer sample and another k-mer in stream.

Streaming Model Considerations

In the previous analysis, we assumed:

- Can access k-mer at any position in $O(1)$ time.

Streaming Model Considerations

In the previous analysis, we assumed:

- Can access k-mer at any position in $O(1)$ time.
- i.e., assumed input strings are stored in something like RAM.

Streaming Model Considerations

In the previous analysis, we assumed:

- Can access k-mer at any position in $O(1)$ time.
- i.e., assumed input strings are stored in something like RAM.

What if strings are sent piece by piece from some source?

Streaming Model Considerations

In the previous analysis, we assumed:

- Can access k-mer at any position in $O(1)$ time.
- i.e., assumed input strings are stored in something like RAM.

What if strings are sent piece by piece from some source?

WLOG, assume each piece is a k-mer.

Streaming Model Considerations

In the previous analysis, we assumed:

- Can access k-mer at any position in $O(1)$ time.
- i.e., assumed input strings are stored in something like RAM.

What if strings are sent piece by piece from some source?

WLOG, assume each piece is a k-mer.

Need: random sample from an incoming stream without knowing how many items there are.

Streaming Model Considerations

In the previous analysis, we assumed:

- Can access k-mer at any position in $O(1)$ time.
- i.e., assumed input strings are stored in something like RAM.

What if strings are sent piece by piece from some source?

WLOG, assume each piece is a k-mer.

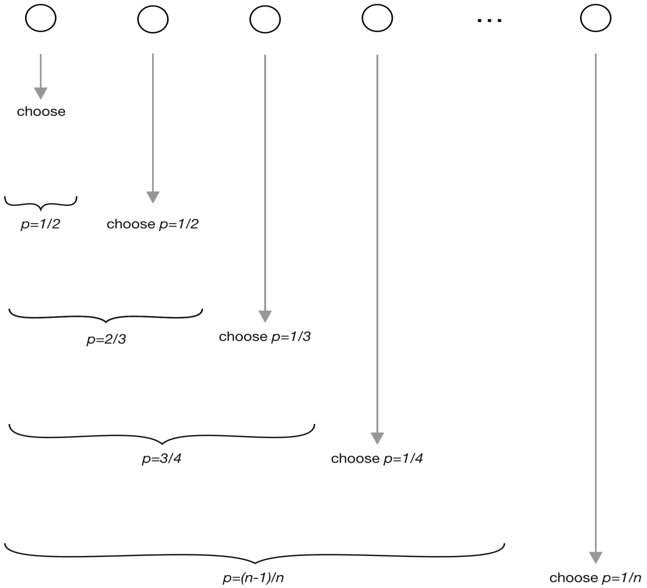
Need: random sample from an incoming stream without knowing how many items there are.

Reservoir Sampling to the rescue.

Streaming Model Considerations

Reservoir Sampling: sample without replacement l items from a stream (unknown length = n).

- We only need to sample 1 item with prob $1/n$.



$$\mathbb{P}[\text{select item } t] = \frac{1}{t} \cdot \frac{t}{t+1} \cdot \frac{t+1}{t+2} \cdots \frac{n-1}{n} = \frac{1}{n}$$

Streaming Model Considerations

Thus, under the incoming stream model, can modify f to be:

Streaming Model Considerations

Thus, under the incoming stream model, can modify f to be:

1. Use Reservoir Sampling to sample a k-mer σ (and its position) uniformly at random.

Streaming Model Considerations

Thus, under the incoming stream model, can modify f to be:

1. Use Reservoir Sampling to sample a k-mer σ (and its position) uniformly at random.
2. Restart the stream again. Count the occurrence number m of σ in the string from which it is sampled.

Streaming Model Considerations

Thus, under the incoming stream model, can modify f to be:

1. Use Reservoir Sampling to sample a k-mer σ (and its position) uniformly at random.
2. Restart the stream again. Count the occurrence number m of σ in the string from which it is sampled.
3. Count whether the other string contains at least m copies of σ, and return 0 or 1 accordingly.

Streaming Model Considerations

Thus, under the incoming stream model, can modify f to be:

1. Use Reservoir Sampling to sample a k-mer σ (and its position) uniformly at random.
2. Restart the stream again. Count the occurrence number m of σ in the string from which it is sampled.
3. Count whether the other string contains at least m copies of σ, and return 0 or 1 accordingly.

- f still has runs in $O(|S_1| + |S_2|)$ time and takes $O(k)$ space.

Streaming Model Considerations

Thus, under the incoming stream model, can modify f to be:

1. Use Reservoir Sampling to sample a k-mer σ (and its position) uniformly at random.
2. Restart the stream again. Count the occurrence number m of σ in the string from which it is sampled.
3. Count whether the other string contains at least m copies of σ, and return 0 or 1 accordingly.

- f still has runs in $O(|S_1| + |S_2|)$ time and takes $O(k)$ space.
- Need 2 passes in total. Can we do it in one pass? (suspect: no)

Outline

Problem Background

My Algorithm for Weighted Jaccard Similarity Estimation

Some other interesting algorithms

Order MinHash [Marcais, 2019]

Assume access to $T_k(S_1), T_k(S_2)$. (all k-mers are labeled with their occurrence numbers).

Order MinHash [Marcais, 2019]

Assume access to $T_k(S_1), T_k(S_2)$. (all k-mers are labeled with their occurrence numbers).

i.e., assume:

$$S_1 = \ldots(\sigma,1)\ldots(\sigma,2)\ldots(\sigma,l)\ldots(\sigma,m)\ldots$$
$$S_2 = \ldots(\sigma,1)\ldots(\sigma,2)\ldots(\sigma,l)\ldots$$

Order MinHash [Marcais, 2019]

Assume access to $T_k(S_1), T_k(S_2)$. (all k-mers are labeled with their occurrence numbers).

i.e., assume:

$$S_1 = \dots(\sigma, 1)\dots(\sigma, 2)\dots(\sigma, l)\dots(\sigma, m)\dots$$
$$S_2 = \dots(\sigma, 1)\dots(\sigma, 2)\dots(\sigma, l)\dots$$

- Estimate (Levenshtein) editing similarity:

$$Ed(S_1, S_2) = \frac{\text{1 - editing distance}}{\max\{|S_1|, |S_2|\}}$$

- Hash functions $\mathcal{H}$: permutations of $T_k(S_1) \cup T_k(S_2)$.

Order MinHash [Marcais, 2019]

Assume access to $T_k(S_1), T_k(S_2)$. (all k-mers are labeled with their occurrence numbers).

i.e., assume:

$$S_1 = \dots(\sigma, 1)\dots(\sigma, 2)\dots(\sigma, l)\dots(\sigma, m)\dots$$
$$S_2 = \dots(\sigma, 1)\dots(\sigma, 2)\dots(\sigma, l)\dots$$

- Estimate (Levenshtein) editing similarity:

$$Ed(S_1, S_2) = \frac{\text{1 - editing distance}}{\max\{|S_1|, |S_2|\}}$$

- Hash functions $\mathcal{H}$: permutations of $T_k(S_1) \cup T_k(S_2)$.
- Sketch: list of l smallest elements in the order of appearance.

Order MinHash [Marcais, 2019]

Assume access to $T_k(S_1), T_k(S_2)$. (all k-mers are labeled with their occurrence numbers).

i.e., assume:

$$S_1 = \ldots(\sigma,1)\ldots(\sigma,2)\ldots(\sigma,l)\ldots(\sigma,m)\ldots$$
$$S_2 = \ldots(\sigma,1)\ldots(\sigma,2)\ldots(\sigma,l)\ldots$$

- Estimate (Levenshtein) editing similarity:

$$Ed(S_1, S_2) = \frac{\text{1 - editing distance}}{\max\{|S_1|, |S_2|\}}$$

- Hash functions $\mathcal{H}$: permutations of $T_k(S_1) \cup T_k(S_2)$.
- Sketch: list of l smallest elements in the order of appearance.
- Collison probability p satisfies:

$$s_1 \le Ed(S_1, S_2) \le s_2 \implies p_1(s_1) \le p \le p_2(s_2)$$

for some $p_1, p_2, s_1, s_2 \in \mathbb{R}$. So by repeated experiments, can estimate p, thus $Ed(S_1, S_2)$.

Order MinHash [Marcais, 2019]

- Editing distance is a very accurate measure.
- But need the stronger assumption about knowing $T_k(S_1), T_k(S_2)$.

HyperLogLog [Flajolet, 2007]

- Estimate number of distinct elements. (Set Cardinality)

HyperLogLog [Flajolet, 2007]

- Estimate number of distinct elements. (Set Cardinality)
- Sketches for A, B can easily be combined to estimate $|A \cup B|$.

HyperLogLog [Flajolet, 2007]

- Estimate number of distinct elements. (Set Cardinality)
- Sketches for A, B can easily be combined to estimate $|A \cup B|$.
- Inclusion-exclusion principle: $|A \cap B| = |A| + |B| - |A \cup B|$.

HyperLogLog [Flajolet, 2007]

- Estimate number of distinct elements. (Set Cardinality)
- Sketches for A, B can easily be combined to estimate $|A \cup B|$.
- Inclusion-exclusion principle: $|A \cap B| = |A| + |B| - |A \cup B|$.
- Can estimate Jaccard index.

HyperLogLog: improvement [Ertl, 2017]

- Same way to obtain sketches: m register values.

HyperLogLog: improvement [Ertl, 2017]

- Same way to obtain sketches: m register values.
- Use a Poisson distribution to model the distribution of the registers.

HyperLogLog: improvement [Ertl, 2017]

- Same way to obtain sketches: m register values.
- Use a Poisson distribution to model the distribution of the registers.
- Turns out Poisson parameter is an unbiased estimator for cardinality.

HyperLogLog: improvement [Ertl, 2017]

- Same way to obtain sketches: m register values.
- Use a Poisson distribution to model the distribution of the registers.
- Turns out Poisson parameter is an unbiased estimator for cardinality.
- Ideas from probabilistic learning: Maximum Likelihood Estimation.

HyperLogLog: improvement [Ertl, 2017]

- Same way to obtain sketches: m register values.
- Use a Poisson distribution to model the distribution of the registers.
- Turns out Poisson parameter is an unbiased estimator for cardinality.
- Ideas from probabilistic learning: Maximum Likelihood Estimation.
- Basically: fit a distribution that is most likely to have produced the observed sketches.

HyperLogLog: improvement [Ertl, 2017]

- Same way to obtain sketches: m register values.
- Use a Poisson distribution to model the distribution of the registers.
- Turns out Poisson parameter is an unbiased estimator for cardinality.
- Ideas from probabilistic learning: Maximum Likelihood Estimation.
- Basically: fit a distribution that is most likely to have produced the observed sketches.

This is implemented by “Dashing” to produce fast and accurate genome distance estimations.

References

[1] *Locality-sensitive hashing for the edit distance. Marcais, 2019*

[2] *New cardinality estimation algorithms for HyperLogLog sketches. Ertl, 2017*

[3] *HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. Flajolet, 2007*

[4] *Dashing: fast and accurate genomic distances with HyperLogLog. Baker & Langmead, 2019*