Contextual Pattern Matching

Gonzalo Navarro

CeBiB — Center for Biotechnology and Bioengineering
Department of Computer Science, University of Chile

- Let's try to find about Einstein's brain in Wikipedia.
- We only find one occurrence, with no much information:

```
During the autopsy, the pathologist of Princeton Hospital, Thomas Stoltz
Harvey, removed Einstein's brain for preservation without the permission of his
family, in the hope that the neuroscience of the future would be able to
discover what made Einstein so intelligent.
```

- Maybe there has been more info in the past about this?
- Searching Pizza&Chili's $\approx$ 6,000 versions of Wikipedia gives us 22,715 matches.
- How are we going to review all those matches, most of them likely identical?

Some grep, sort & hand shows there are only a few different matches:

```
Einstein's brain also contained 73% more glial cells than the average brain.

Einstein's brain also contained 89% more glial cells than the average brain.

Examinations of Albert Einstein's brain after his death have not produced any
conclusive evidence of any particular condition.

..., a groove that normally extends from the front of the brain to the back, did not
go all the way in Einstein's case.

Harvey found nothing unusual with his brain, but in 1999 further analysis by a team
at McMaster University revealed...

Harvey found nothing unusual with his brain, his brain was overall of average size.

..., or the significantly rare and unusual structure of his brain (examined after his
death).

..., the autopsy of Einstein's brain exhibited a more likely possibility that
Einstein, as a child, ...

... believe this may have enabled neurons in this part of his brain a to communicate
better.

The autopsy performed after Einstein's death uncovered anomalies in brain development
which potentially support the theory that...

... Thomas Stoltz Harvey, who removed and preserved Einstein's brain.
```

The problem

- Pattern matching on repetitive text collections
- We have been studying the same problem as always:

  Index a text $T[1..n]$ so that, later, given a pattern $P[1..m]$, return the positions of all the occurrences of P in T.

- But in this case we need some notion of diversity
- What is a reasonable notion at this level of abstraction?
- Can it be implemented efficiently?

Contextual Pattern Matching

The problem

Index a text $T[1..n]$ so that, later, given a pattern $P[1..m]$ and a context length ℓ, return a position in T for each of the c occurrences of distinct strings XPY, with $|X| = |Y| = \ell$.

The challenge

Can we index T in compressed space so as to solve contextual pattern matching in time $\tilde{O}(m + c)$?

Contextual Pattern Matching

The main result (for repetitive collections)

We can use $O(\bar{r} \log(n/\bar{r}))$ space and $O(m + c \log n)$ query time

($\bar{r}$ is the maximum number of runs in the BWT of T and T^{rev})

Other results

We can use $\mathcal{S} + O(n)$ bits and query in time $O(t_s(m) + c\, t_A)$

$\mathcal{S}$ is the space in bits of any index on T^{rev},
$t_s(m)$ is its time to find the suffix array interval and
t_A to access direct and inverse suffix arrays of T and T^{rev}

Contextual Pattern Matching

Linear space

$O(n)$ space and $O(m/\log_\sigma n + c)$ time.

Compact space

$O(n \log \sigma)$ bits and $O(m/\log_\sigma n + (c+1)\log_\sigma^\epsilon n$ time.

Statistically compressed space

$nH_k(T) + o(n \log \sigma) + O(n)$ bits and $O(m + c \log n)$ time.

(results simplified and improved from SPIRE paper, see arXiv version)

General strategy

1. Find range $A'[rs..re]$ of P^{rev} in the suffix array A' of T^{rev}.
2. Partition $[rs..re]$ into maximal consecutive intervals $[rs_i, re_i]$ where the suffixes share their first $m + \ell$ symbols, $P^{rev} X_i^{rev}$.
3. Map each interval $A'[rs_i, re_i]$ to the interval $A[ds_i..de_i]$ corresponding to the suffixes that start with $X_i P$.
4. Partition each interval $A[ds_i..de_i]$ into k_i maximal consecutive subintervals $A[ds_i^j..de_i^j]$ where the suffixes in each subinterval share their first $m + 2\ell$ symbols, $X_i P Y_j$.
5. Report the c subintervals and a position $A[p]$ for each.

Example: $P = \text{a}$, $\ell = 1$, $T = \$\text{alabaralalabarda}\$$, $T^{rev} = \$\text{adrabalalarabala}\$$.

1. Find range $A'[rs..re]$ of P^{rev} in the suffix array A' of T^{rev}.

Suffixes of T^{rev}

$															
a	$														
a	b	a	l	a	$										
a	b	a	l	a	l	a	r	a	b	a	l	a	$		
a	d	r	a	b	a	l	a	l	a	r	a	b	a	l	a	$
a	l	a	$												
a	l	a	l	a	r	a	b	a	l	a	$				
a	l	a	r	a	b	a	l	a	$						
a	r	a	b	a	l	a	$								
b	a	l	a	$											
b	a	l	a	l	a	r	a	b	a	l	a	$			
d	r	a	b	a	l	a	l	a	r	a	b	a	l	a	$
l	a	$													
l	a	l	a	r	a	b	a	l	a	$					
l	a	r	a	b	a	l	a	$							
r	a	b	a	l	a	$									
r	a	b	a	l	a	l	a	r	a	b	a	l	a	$	

Example: $P = \text{a}$, $\ell = 1$, $T = \$\text{alabaralalabarda}\$$, $T^{rev} = \$\text{adrabalalarabala}\$$.

2. Partition $[rs..re]$ into maximal consecutive intervals $[rs_i, re_i]$ where the suffixes share their first $m + \ell$ symbols, $P^{rev} X_i^{rev}$.

Suffixes of T^{rev}

$															
a	$														
a	b	a	l	a	$										
a	b	a	l	a	l	a	r	a	b	a	l	a	$		
a	d	r	a	b	a	l	a	l	a	r	a	b	a	l	a	$
a	l	a	$												
a	l	a	l	a	r	a	b	a	l	a	$				
a	l	a	r	a	b	a	l	a	$						
a	r	a	b	a	l	a	$								
b	a	l	a	$											
b	a	l	a	l	a	r	a	b	a	l	a	$			
d	r	a	b	a	l	a	l	a	r	a	b	a	l	a	$
l	a	$													
l	a	l	a	r	a	b	a	l	a	$					
l	a	r	a	b	a	l	a	$							
r	a	b	a	l	a	$									
r	a	b	a	l	a	l	a	r	a	b	a	l	a	$	

Example: $P = \text{a}$, $\ell = 1$, $T = \$\text{alabaralalabarda}\$$, $T^{rev} = \$\text{adrabalalarabala}\$$.

3. Map each interval $A'[rs_i, re_i]$ to the interval $A[ds_i..de_i]$ corresponding to the suffixes that start with $X_i P$.

Suffixes of T^{rev}

$															
a	$														
a	b	a	l	a	$										
a	b	a	l	a	l	a	r	a	b	a	l	a	$		
a	d	r	a	b	a	l	a	l	a	r	a	b	a	l	a	$
a	**l**	a	$												
a	**l**	a	l	a	r	a	b	a	l	a	$				
a	**l**	a	r	a	b	a	l	a	$						
a	r	a	b	a	l	a	$								
b	a	l	a	$											
b	a	l	a	l	a	r	a	b	a	l	a	$			
d	r	a	b	a	l	a	l	a	r	a	b	a	l	a	$
l	a	$													
l	a	l	a	r	a	b	a	l	a	$					
l	a	r	a	b	a	l	a	$							
r	a	b	a	l	a	$									
r	a	b	a	l	a	l	a	r	a	b	a	l	a	$	

Example: $P = \text{a}$, $\ell = 1$, $T = \$\text{alabaralalabarda}\$$, $T^{rev} = \$\text{adrabalalarabala}\$$.

3. Map each interval $A'[rs_i, re_i]$ to the interval $A[ds_i..de_i]$ corresponding to the suffixes that start with $X_i P$.

Suffixes of T

$															
a	$														
a	b	a	r	a	l	a	l	a	b	a	r	d	a	$	
a	b	a	r	d	a	$									
a	l	a	b	a	r	a	l	a	l	a	b	a	r	d	a	$
a	l	a	b	a	r	d	a	$							
a	l	a	l	a	b	a	r	d	a	$					
a	r	a	l	a	l	a	b	a	r	d	a	$			
a	r	d	a	$											
b	a	r	a	l	a	l	a	b	a	r	d	a	$		
b	a	r	d	a	$										
d	a	$													
l	**a**	b	a	r	a	l	a	l	a	b	a	r	d	a	$
l	**a**	b	a	r	d	a	$								
l	**a**	l	a	b	a	r	d	a	$						
r	a	l	a	l	a	b	a	r	d	a	$				
r	d	a	$												

Example: $P = \text{a}$, $\ell = 1$, $T = \$\text{alabaralalabarda}\$$, $T^{rev} = \$\text{adrabalalarabala}\$$.

4. Partition each interval $A[ds_i..de_i]$ into k_i maximal consecutive subintervals $A[ds_i^j..de_i^j]$ where the suffixes in each subinterval share their first $m + 2\ell$ symbols, $X_i P Y_j$.

Suffixes of T

$															
a	$														
a	b	a	r	a	l	a	l	a	b	a	r	d	a	$	
a	b	a	r	d	a	$									
a	l	a	b	a	r	a	l	a	l	a	b	a	r	d	a	$
a	l	a	b	a	r	d	a	$								
a	l	a	l	a	b	a	r	d	a	$						
a	r	a	l	a	l	a	b	a	r	d	a	$				
a	r	d	a	$												
b	a	r	a	l	a	l	a	b	a	r	d	a	$			
b	a	r	d	a	$											
d	a	$														
l	a	b	a	r	a	l	a	l	a	b	a	r	d	a	$	
l	a	b	a	r	d	a	$									
l	a	l	a	b	a	r	d	a	$							
r	a	l	a	l	a	b	a	r	d	a	$					
r	d	a	$													

Example: $P = \text{a}$, $\ell = 1$, $T = \$\text{alabaralalabarda}\$$, $T^{rev} = \$\text{adrabalalarabala}\$$.

5. Report the c subintervals and a position $A[p]$ for each.

A	Suffixes of T
17	**$**
16	a $
3	a b a r a l a l a b a r d a $
11	a b a r d a $
1	a l a b a r a l a l a b a r d a $
9	a l a b a r d a $
7	a l a l a b a r d a $
5	a r a l a l a b a r d a $
13	a r d a $
4	**b a r** a l a l a b a r d a $
12	b a r d a $
15	**d a $**
2	**l a b** a r a l a l a b a r d a $
10	l a b a r d a $
8	**l a l** a b a r d a $
6	**r a l** a l a b a r d a $
14	r d a $

$\$\text{alabaralalabarda}\$$

Tools for repetitive collections

Gagie, N. & Prezza [JACM 2020] show how to build $O(r \log(n/r))$-space data structures on $T[1..n]$, with r BWT runs, suffix array A and LCP array LCP, so as to compute:

- The range of $P[1..m]$ in A, in time $O(m)$.
- Any entry of A, A^{-1}, and LCP, in time $O(\log(n/r))$.
- Further queries in time within $O(\log n)$:
  - $RMQ(i,j) = \text{argmin}_{i \le k \le j} LCP[k]$.
  - $PSV(p,d) = \max(\{q < p, LCP[q] < d\} \cup \{0\})$.
  - $NSV(p,d) = \min(\{q > p, LCP[q] < d\} \cup \{n+1\})$.

2. Partition $[rs..re]$ into maximal consecutive intervals $[rs_i, re_i]$ where the suffixes share their first $m + \ell$ symbols, $P^{rev} X_i^{rev}$.

$\longrightarrow$ Find all values $LCP'[p] < m + \ell$ using RMQs.

LCP'	Suffixes of T^{rev}																
0	$																
0	**a**	$															
0	**a**	b	a	l	a	$											
5	**a**	b	a	l	a	l	a	r	a	b	a	l	a	$			
1	**a**	d	r	a	b	a	l	a	l	a	r	a	b	a	l	a	$
1	**a**	l	a	$													
3	**a**	l	a	l	a	r	a	b	a	l	a	$					
3	**a**	l	a	r	a	b	a	l	a	$							
1	**a**	r	a	b	a	l	a	$									
0	b	a	l	a	$												
4	b	a	l	a	l	a	r	a	b	a	l	a	$				
0	d	r	a	b	a	l	a	l	a	r	a	b	a	l	a	$	
0	l	a	$														
2	l	a	l	a	r	a	b	a	l	a	$						
2	l	a	r	a	b	a	l	a	$								
0	r	a	b	a	l	a	$										
6	r	a	b	a	l	a	l	a	r	a	b	a	l	a	$		

3. Map each interval $A'[rs_i, re_i]$ to the interval $A[ds_i..de_i]$ corresponding to the suffixes that start with X_iP.

$\longrightarrow$ We map any $A'[p']$ to $A[p]$: $p = A^{-1}[n - A'[p'] - (m + \ell - 1)]$

$\longrightarrow$ We then extend p up and down with *PSV* and *NSV* looking for $LCP < m + \ell$.

LCP	Suffixes of T																
0	$																
0	a	$															
1	a	b	a	r	a	l	a	l	a	b	a	r	d	a	$		
4	a	b	a	r	d	a	$										
1	a	l	a	b	a	r	a	l	a	l	a	b	a	r	d	a	$
6	a	l	a	b	a	r	d	a	$								
3	a	l	a	l	a	b	a	r	d	a	$						
1	a	r	a	l	a	l	a	b	a	r	d	a	$				
2	a	r	d	a	$												
0	b	a	r	a	l	a	l	a	b	a	r	d	a	$			
3	b	a	r	d	a	$											
0	d	a	$														
0	l	a	b	a	r	a	l	a	l	a	b	a	r	d	a	$	
5	l	a	b	a	r	d	a	$									
2	l	a	l	a	b	a	r	d	a	$							
0	r	a	l	a	l	a	b	a	r	d	a	$					
1	r	d	a	$													

Other indexes

- We need $t_s(m)$ time to find the first interval
- We need $c \cdot t_A$ time to compute various A, A', A^{-1} entries
- We use $O(n)$ bits to support the RMQs in constant time
- Instead of PSV and NSV we define $C[i] = A^{-1}[n - A'[i]]$ and build RMQ_C, so

$$\begin{aligned} ds_i &= A^{-1}[n - A'[RMQ_C(rs_i, re_i)]] - (m + \ell - 1) \\ de_i &= ds_i + (re_i - rs_i) \end{aligned}$$

- Thus we add $O(n)$ bits and no asymptotic time on top of $O(t_s(m) + c\, t_A)$.

Conclusions

- New search problem, natural for repetitive collections.
- Analogous to the diversified search concept in IR.
- We obtained satisfactory time/space tradeoffs...
- ... except, perhaps, precisely the space on repetitive texts: $O(\overline{r}\log(n/\overline{r}))$ is not that nice.
- Can we do it within $O(\overline{r})$ space?
- Can we do it over locally consistent grammars?
- Other natural queries for repetitive text collections:
  - Document retrieval queries in general.
  - Range-restricted or subtree-restricted queries.
  - Range-collapsed document listing.
  - Granularity-bounded document listing.
  - Others?