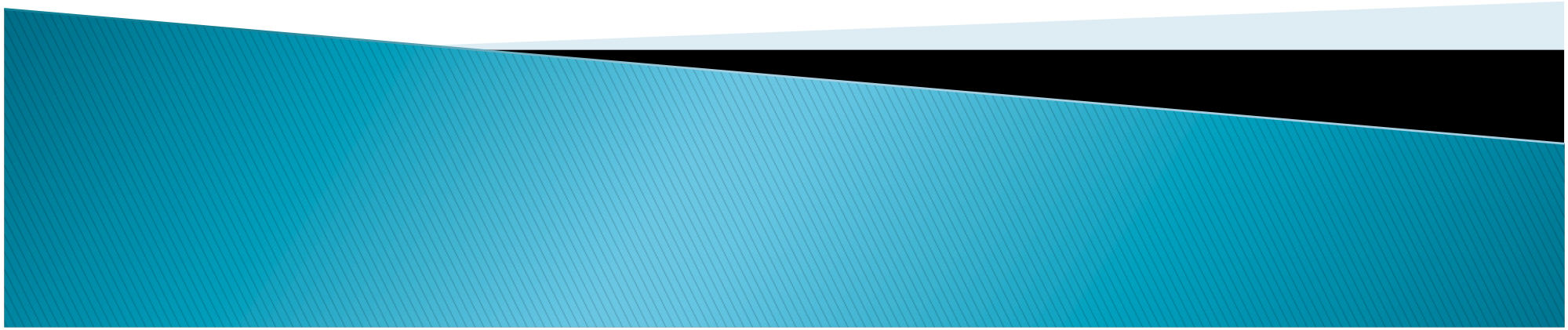
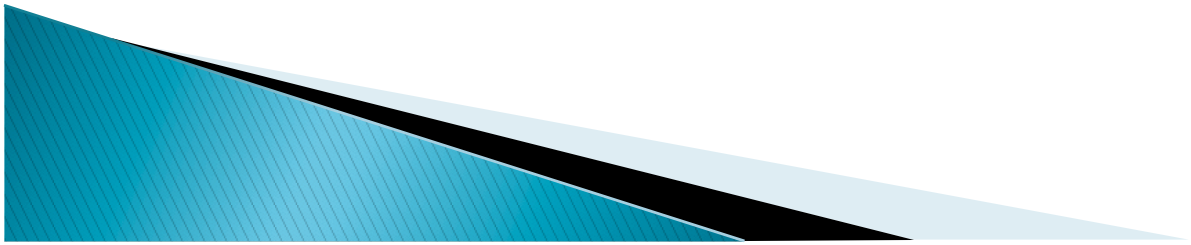
String and Tree Kernels Algorithms and Applications

Devendra Chaplot (100050033)



NOTATIONS

- **Alphabet**
  Set of characters denoted by A
- **String**
  Any $x \in A^k$ for some k
- **Sentinel**
  Some $\$ \notin A$ used to terminate a string
- **Prefix/Suffix/Substring**
  Let x = uvw for some possibly empty u, v and w,
  u is called the prefix,
  w the suffix of x, v is called a substring of x
  We write $v \sqsubseteq x$

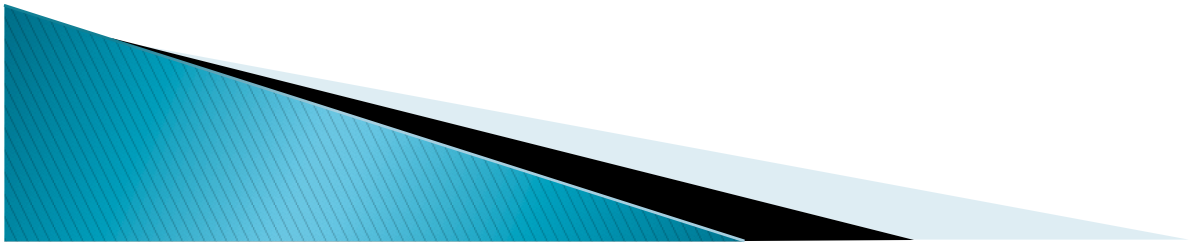


String Kernels

- **Exact Matching Kernels**

$$k(x, x') := \sum_{s \sqsubseteq x, s' \sqsubseteq x'} w_s \delta_{s,s'} = \sum_{s \in \mathcal{A}^*} \mathrm{num}_s(x)\, \mathrm{num}_s(x') w_s.$$

- Count all matching substrings
- Flexible weighting schemes $\in A*$
- Different applications ⇒ different weights
- Noise in training data ⇒ does not work :-(
- Successful applications in bio-informatics (Vishwanathan and Smola, 2002) (Leslie et. al., 2002)
- Linear time algorithms using suffix trees



Exact String Kernels

- **Bag of Characters**

  Counts single characters. Set $w_s = 0$ for all $|s| > 1$

- **Bag of Words**

  s is bounded by whitespace (Joachims, 1999)

- **Limited Range Correlations**

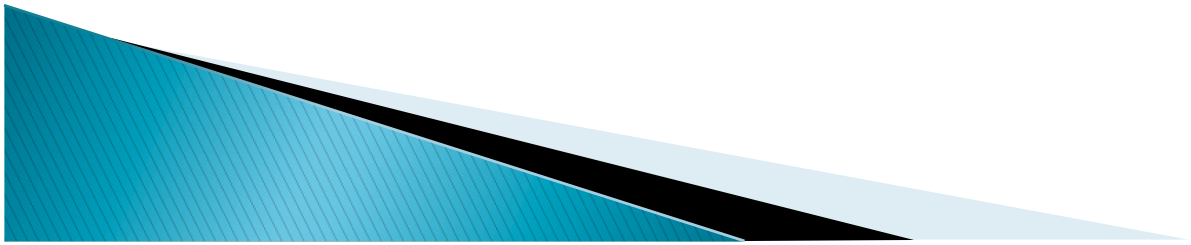
  Set $w_s = 0$ for all $|s|>n$ given a fixed n

- **K-spectrum kernel**

  Account for matching substrings of length k (Leslie et al., 2002). Setws =0for all$|s|\neq k$

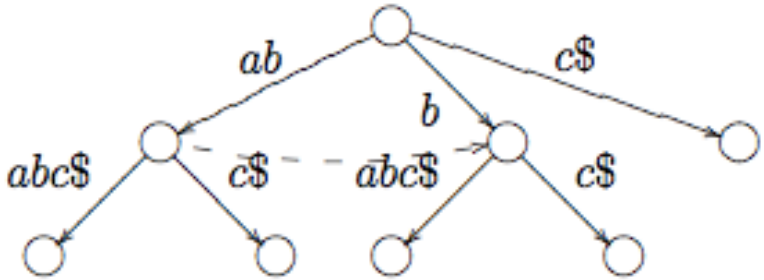
- **General Case**

  Quadratic time kernel computation (Haussler, 1998, Watkins, 1998), cubic time prediction



Suffix Trees

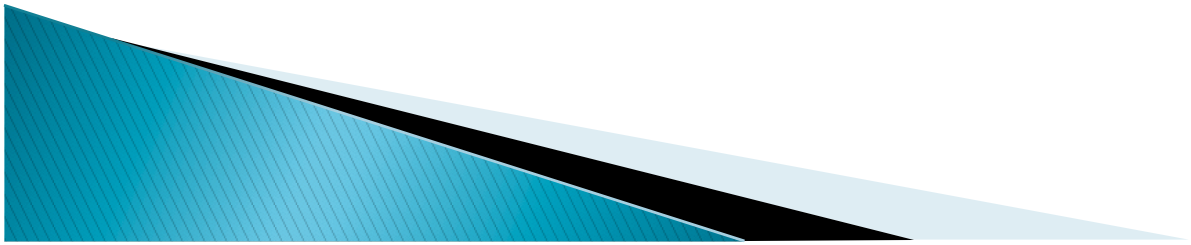
- **Definition**
  - Compact tree built from all the suffixes of a word. Suffix tree of *ababc* denoted by *S(ababc)*.



  - Node label := unique path from the root
  - Suffix links are used to speed up parsing of strings
  - Suppose we are at a node ax then suffix links help us to jump to node x
  - Each internal node has a unique suffix link (McCreight, 76)

Properties of Suffix Trees

- **Represents all the substrings of the given string**
- **Can be constructed in linear time**
- **Offline algorithms - (Weiner 73) and (McCreight 76)**
- **Online algorithm - (Ukkonen 93)**
- **Can be stored using linear space**
- **Edges are encoded by indices of the substring**
- **Each leaf corresponds to a unique suffix**
- **Leaves on subtree give number of occurrences**
- **Each internal node has at least 2 distinct children**



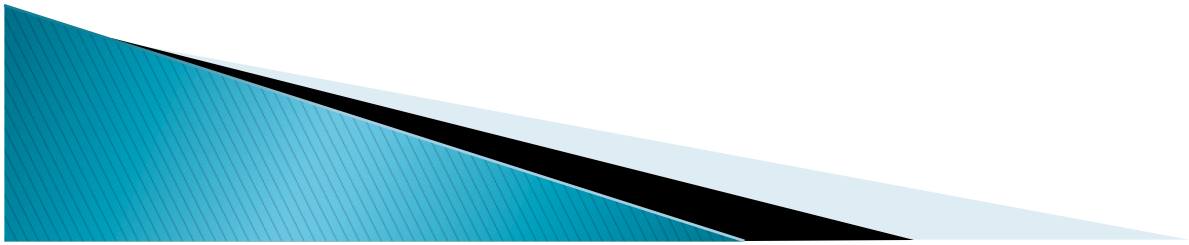
Matching Statistics

- **Definition**

  Given strings x, y with $|x| = n$ and $|y| = m$, the matching statistics of x with respect to y are defined by $v, c \in \mathbb{N}^n$, where

  - v_i is the length of the longest substring of y matching a prefix of $x[i : n]$
  - $\overline{v_i} := i + v_i - 1$
  - c_i is a pointer to $\text{ceil}(x[i : \overline{v_i}])$ in $S(y)$.
  - $\text{ceil}(s)$ is the last node on the path from the root to s

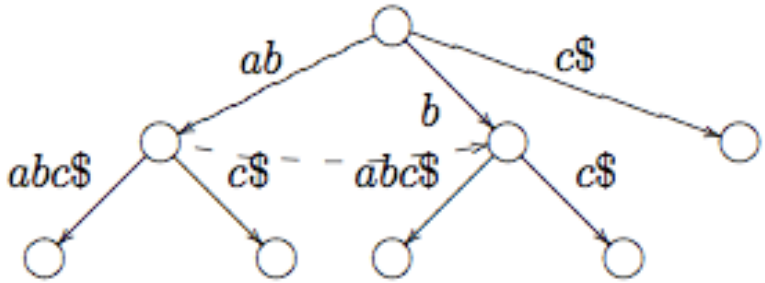
  This can be computed in linear time (Chang and Lawler, 1994).



Example

- Matching statistic of abba with respect to S(ababc)

String	a	b	b	a
v_i	2	1	2	1
ceil(c_i)	ab	b	b	root



Matching Substrings

- **Prefixes**

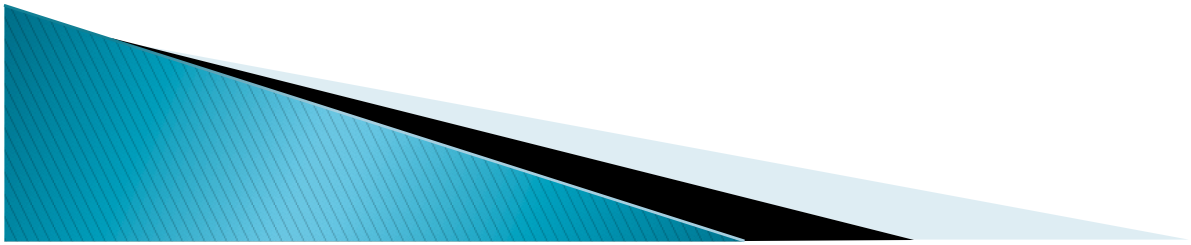
  w is a substring of x iff there is an i such that w is a prefix of $x[i : n]$. The number of occurrences of w in x can be calculated by finding all such i.

- **Substrings**

  The set of matching substrings of x and y is the set of all prefixes of $x[i : \overline{v_i}]$.

- **Next Step**

  If we have a substring w of x, prefixes of w may occur in x with higher frequency. We need an efficient computation scheme.



Key Concept

Theorem

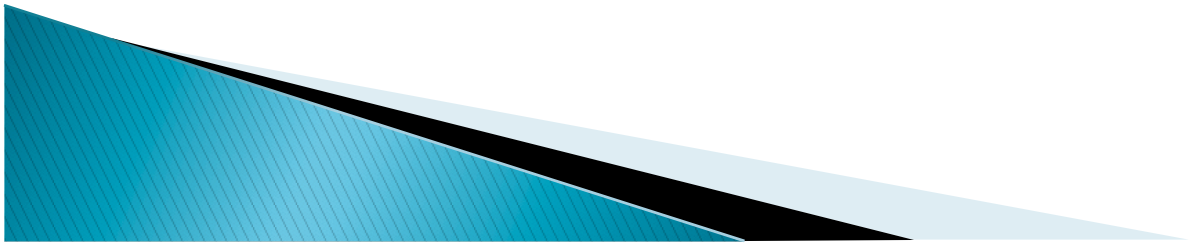
Let x and y be strings and c and v be the matching statistics of x with respect to y. Assume that

$$W(y,t) = \sum_{s \in \text{prefix}(v)} w_{us} - w_u \text{ where } u = \text{ceil}(t) \text{ and } t = uv.$$

can be computed in constant time for any t. Then $k(x,y)$ can be computed in $O(|x| + |y|)$ time as

$$k(x,y) = \sum_{i=1}^{|x|} \text{val}(x[i : \overline{v_i}])$$

where $\text{val}(s)$ indicates the contribution to the kernel due to string s and all its prefixes.



Key Concept

- **Observation**

  All substrings ending on the same edge of $s(y)$ occur the same number of times in string y

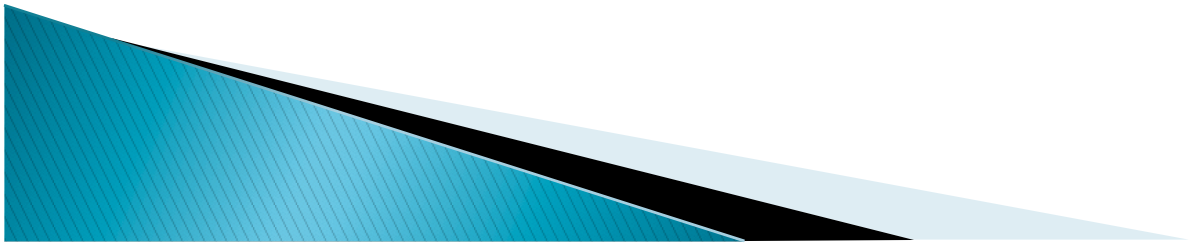
  For any matching substring t we can write

$$\mathrm{val}(t) := \mathrm{lvs}(\mathrm{floor}(t)) \cdot W(y,t) + \mathrm{val}(\mathrm{ceil}(t))$$

  For each node v we can pre-compute $\mathrm{val}(v)$ by a simple DFS on the suffix tree

  We can compute in constant time

$$\mathrm{val}(x[i:\overline{v_i}]) = \mathrm{lvs}(\mathrm{floor}(x[i:\overline{v_i}])) \cdot W(y, x[i:\overline{v_i}]) + \mathrm{val}(c_i)$$



Computations

Length-Dependent Weights

Assume that $w_s = w_{|s|}$, then

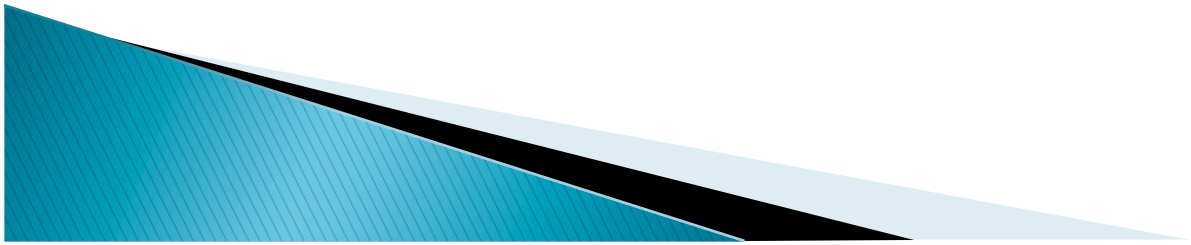
$$W(y,t) = \sum_{j=|\operatorname{ceil}(t)|}^{|t|} w_j - w_{|\operatorname{ceil}(t)|} = \omega_{|t|} - \omega_{|\operatorname{ceil}(t)|}$$

where $\omega_j := \sum_{i=1}^{j} w_j$, which can be pre-computed and stored for $j = 1, 2, \ldots \max(|x|, |y|)$.

K-spectrum Kernel

It is easy to see that

$$W(y,t) = \begin{cases} 1 & \text{if } |\operatorname{ceil}(t)| < k \text{ and } |t| \geq k \\ 0 & \text{if otherwise} \end{cases}$$



Tree Kernels

Subset Trees

Set of connected nodes of a tree T

Definition (Colins and Duffy, 2001)

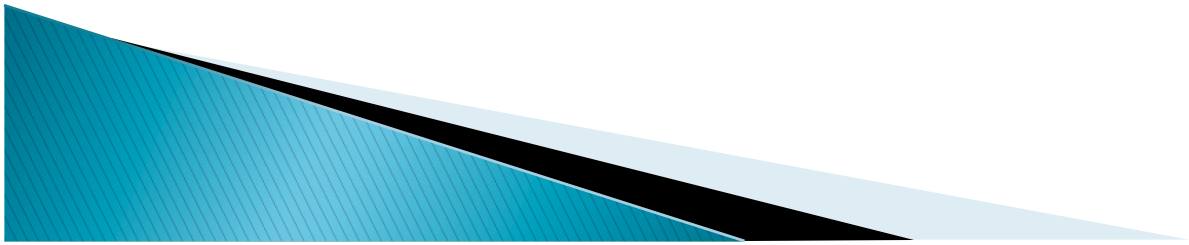
Denote by T, T' trees and by $t \models T$ a subset tree of T, then

$$k(T, T') = \sum_{t \models T, t' \models T'} w_t \delta_{t,t'}.$$

Our Definition (Vishwanathan and Smola, 2002)

In case we count matching subtrees then $t \models T$ denotes that t is a subtree of T and we get

$$k(T, T') = \sum_{t \models T, t' \models T'} w_t \delta_{t,t'}.$$



Permutation Invariance

- **Problem**

  We want permutation invariance of unordered trees.

- **Example:**

  The following two unordered trees are mirror images

- **Solution**

  Sort trees before computing kernel

  Maps equivalent trees to a single representative

Sorting Trees

Sorting Rules

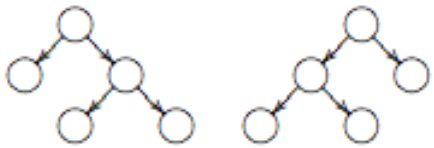
- Assume existence of lexicographic order on labels
- Introduce symbols '[', ']' satisfy '[' < ']', and that ']', '[' < $\mathrm{label}(n)$ for all labels.

Algorithm

- For node n with children $n_1, \ldots, n_c$ sort the tags of the children in lexicographical order such that $\mathrm{tag}(n_i) \leq \mathrm{tag}(n_j)$ if $i < j$ and define

$$\mathrm{tag}(n) = [\,\mathrm{label}(n)\,\mathrm{tag}(n_1)\,\mathrm{tag}(n_2)\ldots\mathrm{tag}(n_c)]\,.$$

Example

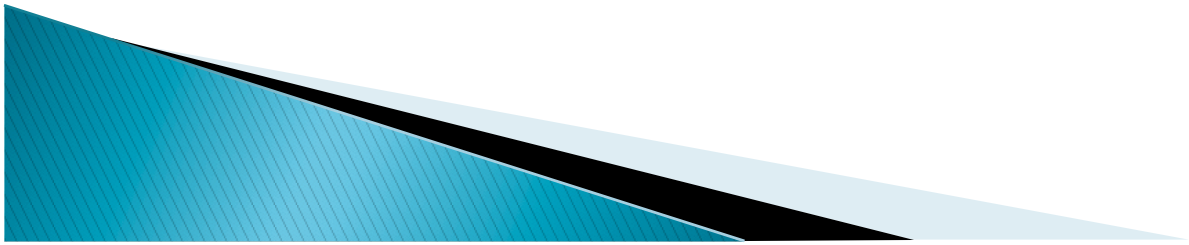
The trees  have label `[[][[][]]]`.

Sorting trees

Theorem

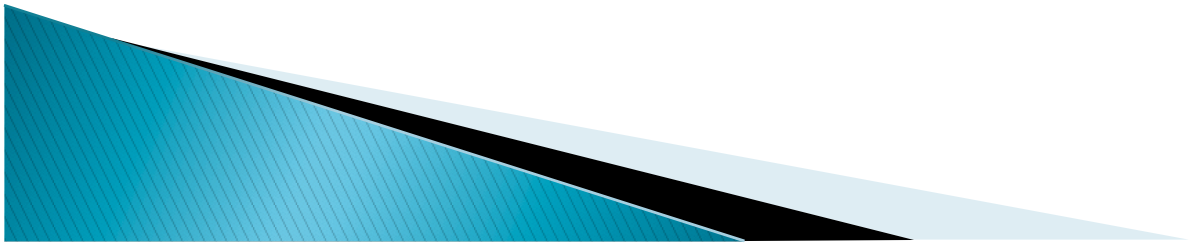
Let l be the number of nodes and λ the length of a label

1. $\mathrm{tag}(\mathrm{root})$ can be computed in $(\lambda + 2)(l \log_2 l)$ time and linear storage in l.
2. Substrings s of $\mathrm{tag}(\mathrm{root})$ starting with '$[$' and ending with a balanced '$]$' correspond to subtrees t of T where s is the tag on t.
3. $\mathrm{tag}(\mathrm{root})$ is invariant under permutations of the leaves and allows the reconstruction of an unique element of the equivalence class (under permutation).



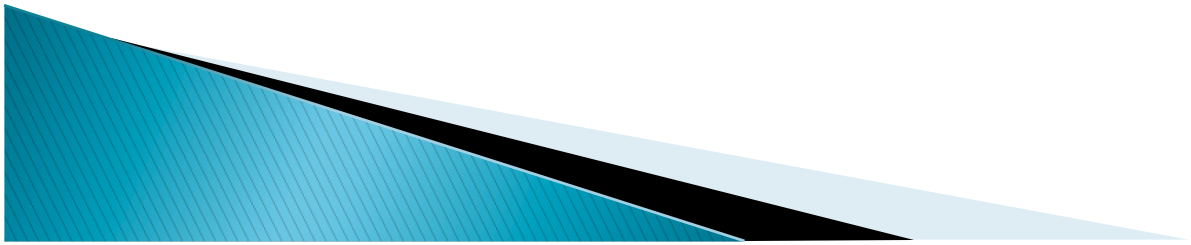
Testing: String Kernel

- Data set used: **Molecular Biology (Splice-junction Gene Sequences) Data Set**
  - *http://archive.ics.uci.edu/ml/datasets/Molecular+Biology+(Splice-junction+Gene+Sequences)*
- Abstract: Primate splice-junction gene sequences (DNA) with associated imperfect domain theory
- Number of Instances = 3190
- Number of Attributes = 61
- Number of classes = 3
- Used Exact String Kernel and length dependent weights



Observations (1)

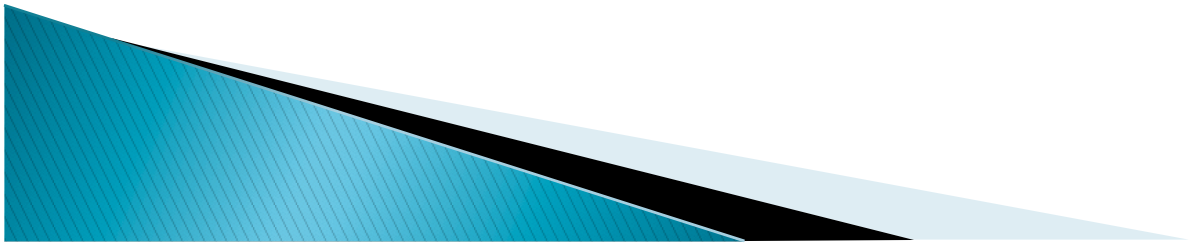
Training Size	Test Size	Average Accuracy
10	20	30%
20	20	42%
50	20	57%
100	20	62.5%
200	20	69.5%
500	20	70%



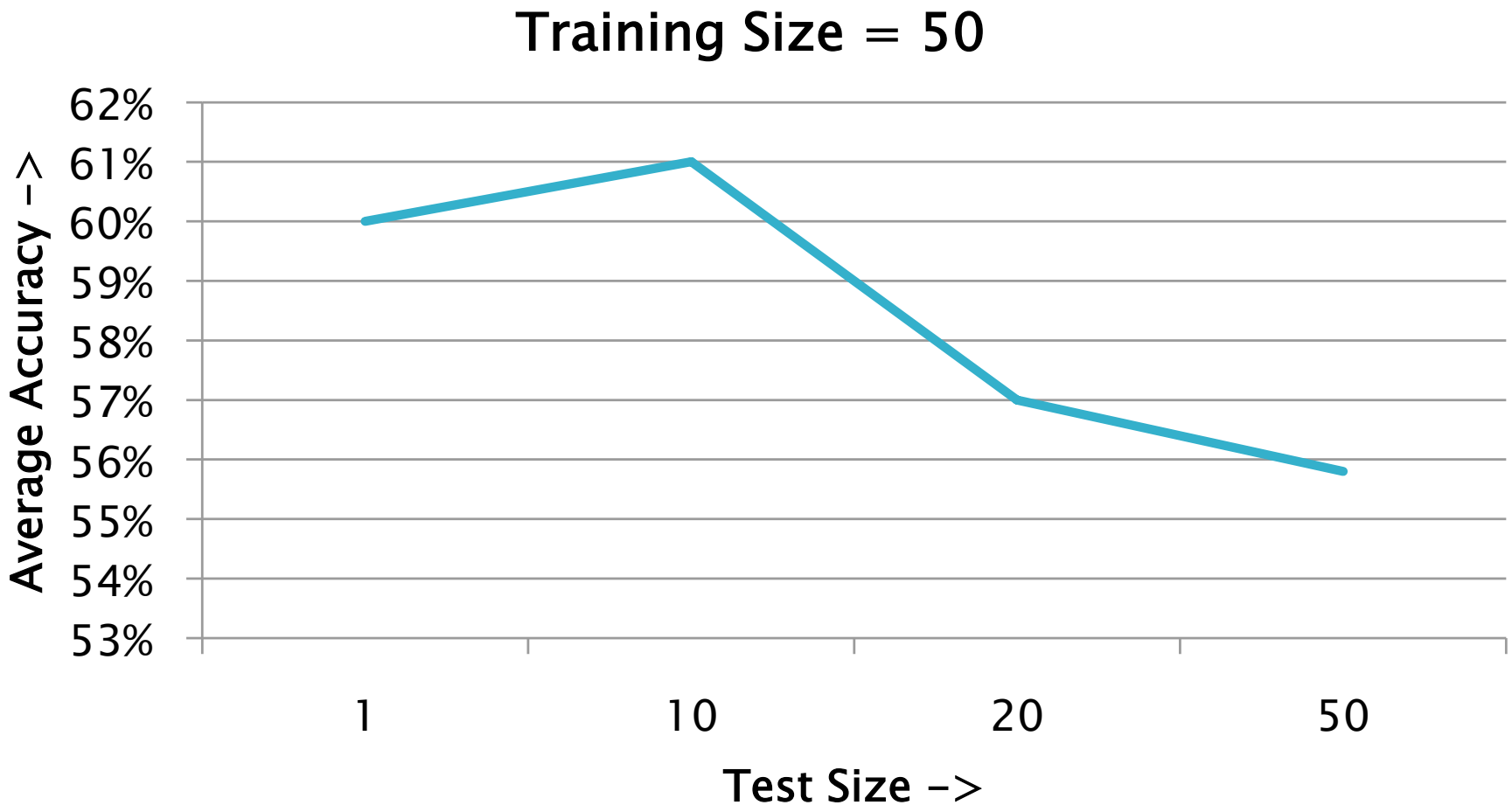
Training Size vs Average Accuracy

Observations (2)

Training Size	Test Size	Average Accuracy
50	1	60%
50	10	61%
50	20	57%
50	50	55.8%

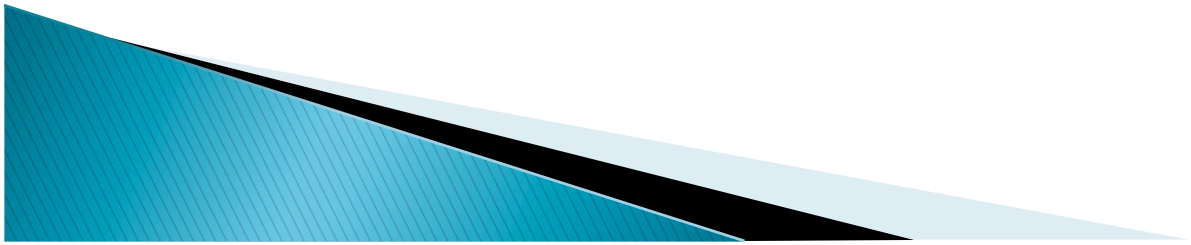


Test Size vs Average Accuracy



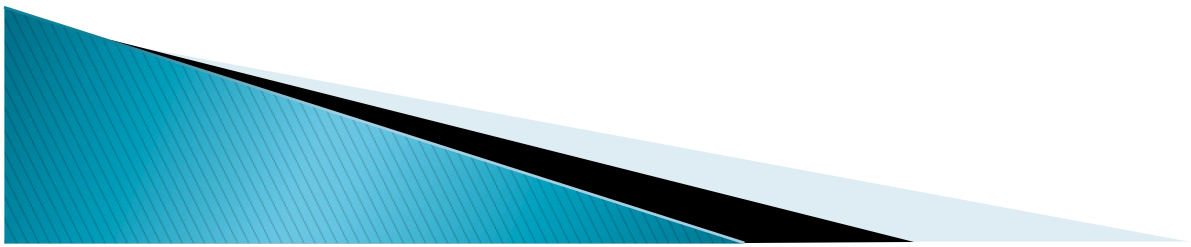
Observations (3)

Training Size	Test Size	Average Accuracy
10	1	30%
100	10	54%
200	20	55%
500	50	59.4%
1000	100	67.7

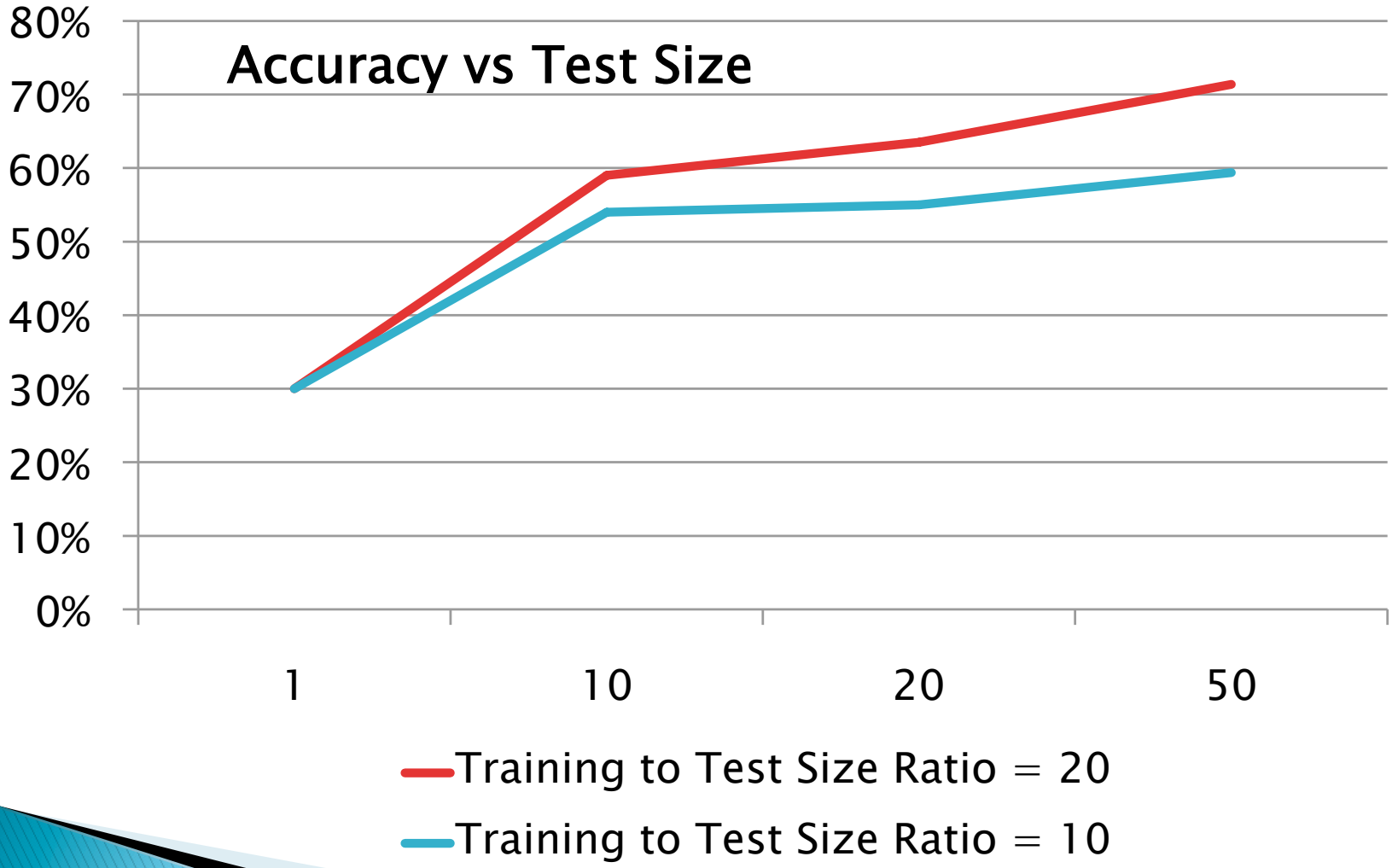


Observations (4)

Training Size	Test Size	Average Accuracy
20	1	30%
200	10	59%
400	20	63.5%
1000	50	71.4%

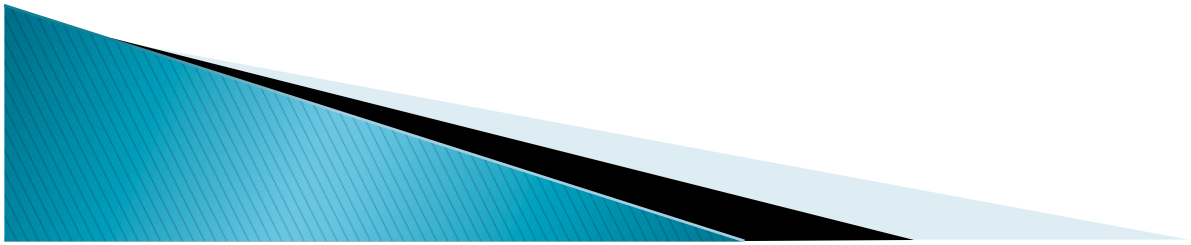


Observations(3-4)



Results

- The accuracy of String Kernel increases as the size of Training Set increases
- The accuracy of String Kernel also increases if the size of training set increases as compared to size of test set
- With sufficient training, string kernel is about 70% accurate
- The accuracy drops as we increase the size of test set keeping size of training set constant



References

- String and Tree Kernels Algorithms and Applications, S.V.N. “Vishy” Vishwanathan
  - http://www.stat.purdue.edu/~vishy/talks/StringKernels.pdf
- Data set used: **Molecular Biology (Splice-junction Gene Sequences) Data Set**
  - *http://archive.ics.uci.edu/ml/datasets/Molecular+Biology+(Splice-junction+Gene+Sequences)*
- LIBSVM(v3.6) – A Library for Support Vector Machines
  - http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- ANSI C implementation of a Suffix Tree
  - http://mila.cs.technion.ac.il/~yona/suffix_tree/