Large-scale Kernel Methods and Algorithms

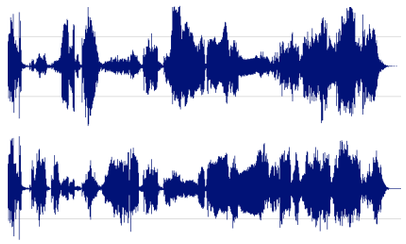
Pavel P. Kuksa
Department of Computer Science
Rutgers University

Joint work with
Vladimir Pavlovic, Jason Weston, Yanjun Qi, Bing Bai,
Ronan Collobert

What are we after?

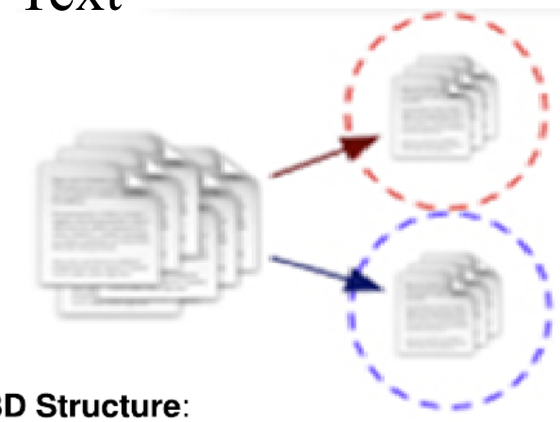
- Scalable methods for annotation, information extraction for structured/sequence data
  - sequence ID (classification): e.g. topic, genre, etc
  - sequence Labeling: e.g. entity tagging
  - information extraction: eg. entity relationships

Audio/Music



Music Genre
Artist
etc

Text



Bio-informatics

Sequence

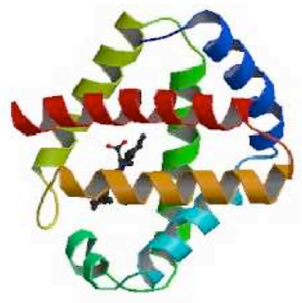
VDAAVAKVCGSEAIKANLRRSWGVLSADIEA
TGLMLMSNLFTLRPDTKTYFTRLGDVQKGK
ANSKLRGHAITLTYALNNFVDSLDDPSRLKC
VVEKFAVNHINRKISGDAFGAIVEPMKETLKA
RMGNYYSDDVAGAWAALVGVVQAAL

predict

Class:
Globin-like

Function:
Oxygen transport

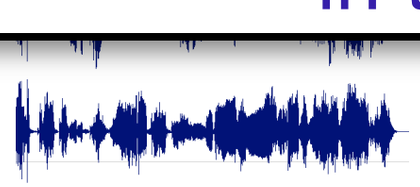
3D Structure:



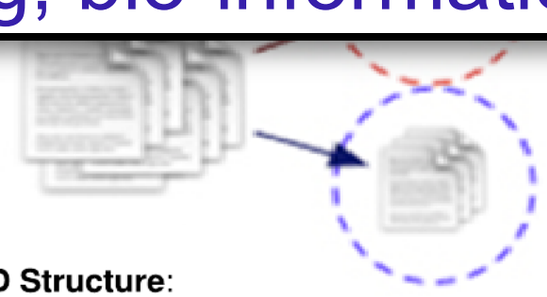
What are we after?

- Scalable methods for annotation, information extraction for structured/sequence data
  - sequence ID (classification): e.g. topic, genre, etc

This talk: (1) New algorithmic framework for general kernel computation on sequence data
(2) Important and challenging practical problems in text, audio mining, bio-informatics



Artist
etc



Bio-informatics

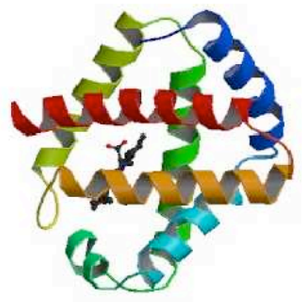
Sequence
VDAAVAKVCGSEAIKANLRRSWGVLSADIEA
TGLMLMSNLFTLRPDTKTYFTRLGDVQKGK
ANSKLRGHAITLTYALNNFVDSLDDPSRLKC
VVEKFAVNHINRKISGDAFGAIVEPMKETLKA
RMGNYYSDDVAGAWAALVGVVQAAL

predict

Class:
Globin-like

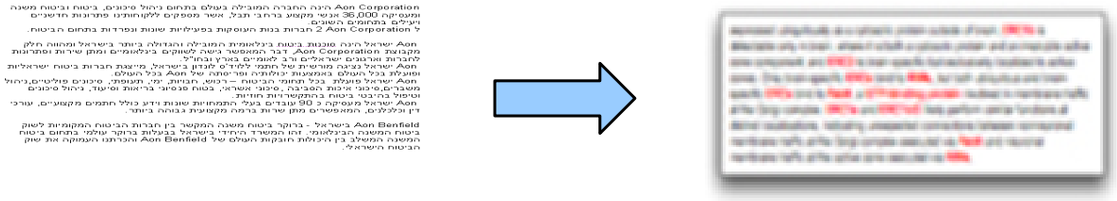
Function:
Oxygen transport

3D Structure:

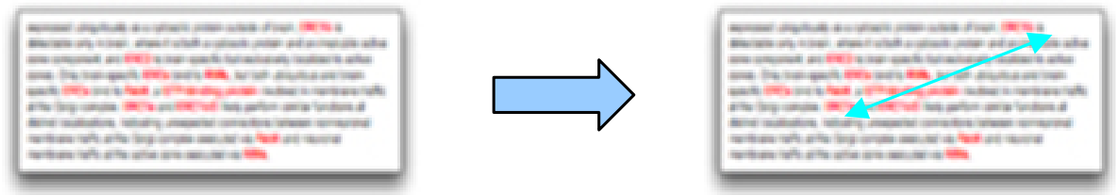


Case I: NLP

- Entity tagging



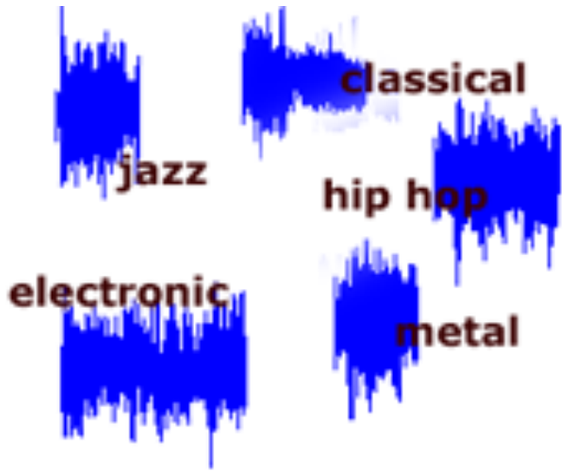
- Relationship extraction



- Relevant document retrieval

Case II: Music ID

- How to tell apart/predict music genre?
- How to tell apart/predict music artists?
- How to tell apart/predict mood or emotions in music?

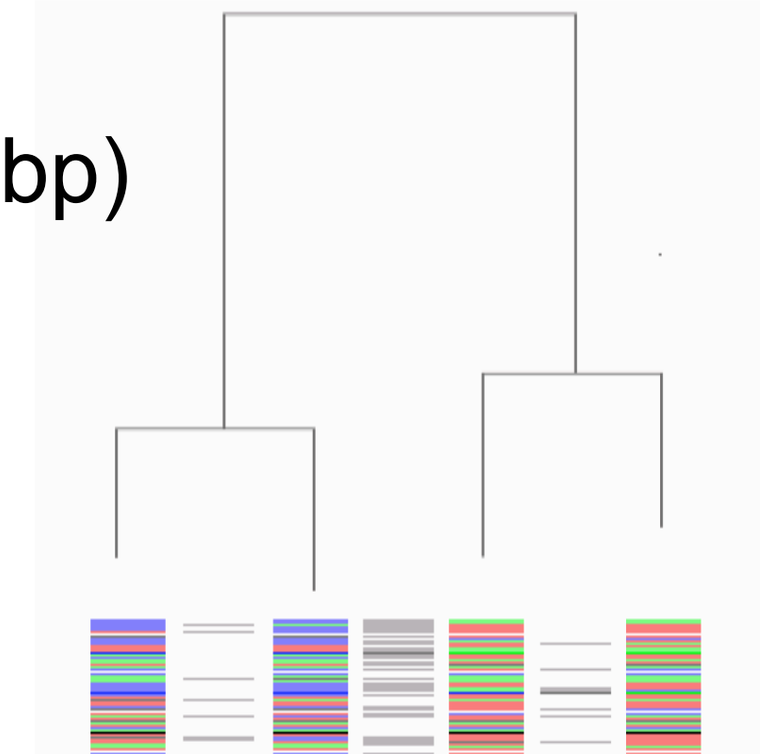


- Challenges:
  - what are important characteristics (features)?
  - great variability in content

Case III: DNA Barcoding

- DNA barcode: a “short” sequence (≈600bp) of mitochondrial DNA (CO1)
- Barcoding of Life:

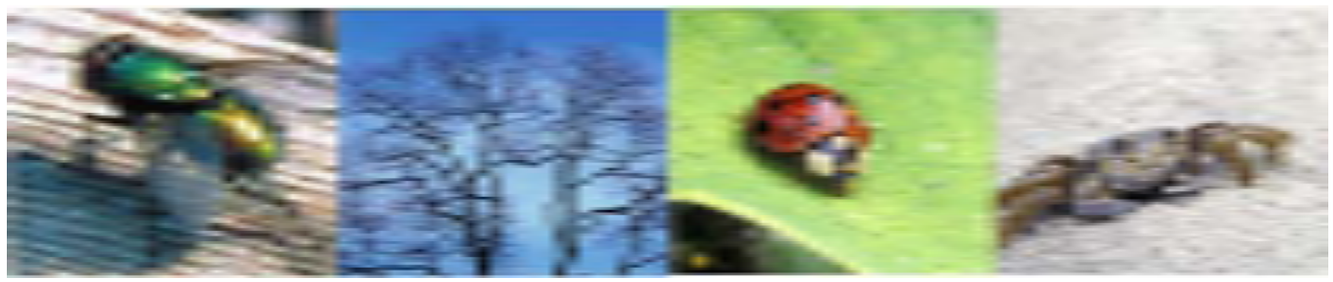
classification and identification of organisms using ‘DNA barcodes’

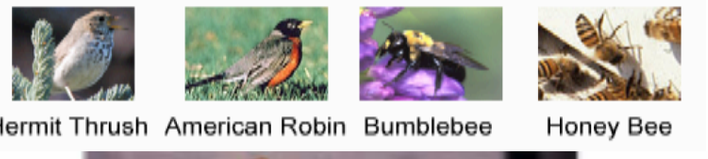


Inherently large-scale problem:

Known species:

>150,000	>250,000	>300,000	>30,000
flies, mosquitoes	flowering plants	beetles	crabs, lobsters

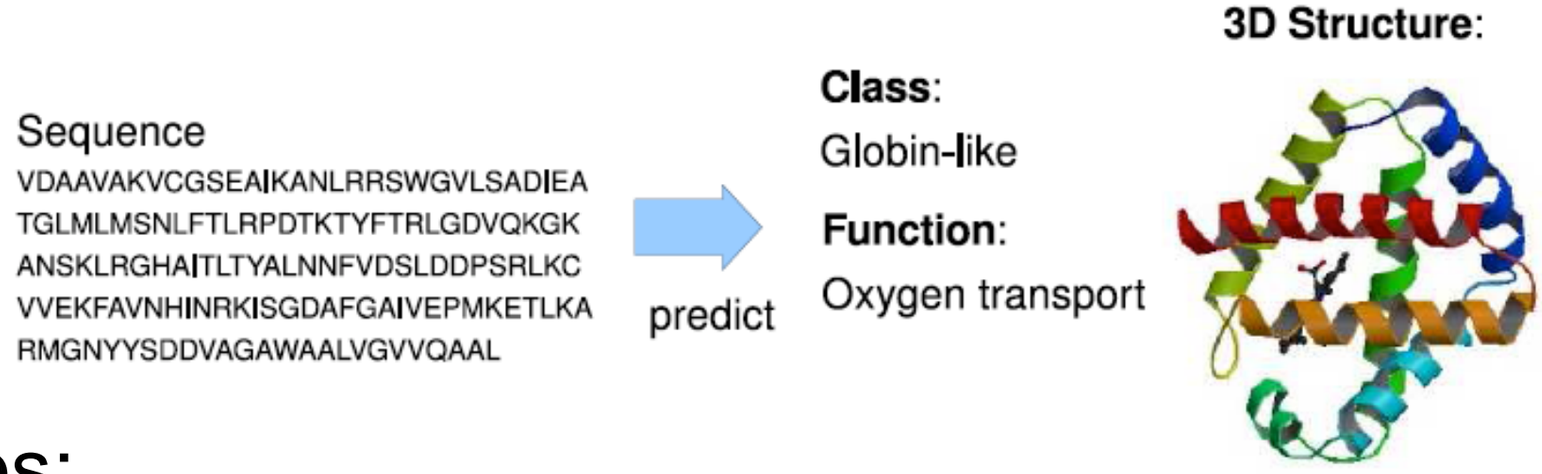




- Applications: taxonomy, biodiversity understanding, invasive species tracking, etc

Case IV: Protein Annotation

- Primary (amino-acid) sequence easy to obtain
- Structural/Functional information difficult to obtain
- **Goal**: predict structural/functional class from primary sequence



- Challenges:
  - Limited labeled data, highly diverse sequences
  - Millions of unannotated sequences, many classes

Solving Sequence ID Problems

Sequence X

GTATATGATCAGGACTAGTTGGTACAGCTCTTAGACTATTAATT
CGAGCTGAACTAGGCCAACCAGGAGCTCTTCTGGGGGATGA
TCAATTATATAATGTAATTGTAACCGCCCACGCTTTTGTTATAAT
TTTCTTTTTAGTAATACCTATAATGATTGGAGGATTTGGAAACT
GATTAGTTCCTTTAATACTAGGAGCCCCAGACATGGCATTCCC
ACGTCTAAATAATATAAGTTTCTGACTTCTTCCGCCTGCCCTT
CTACTACTCCTTTCTTCAGCAGCAGTCGAAAGTGGAGTTGGA
ACTGGATGAACCGTCTACCCTCCTTTAGCCGGAAACCTTGCT
CACGCATTGACAGACCGAAACTTTAACACCGCTTTCTTTGAT
CCAGCAGGAGGTGGAGA

?

Sequence Y

ATGGTTAAATTCTCCTCAGAACCACATTTGGCTCAGGTAGTC
GCAGAAGACCTTCTTTCTCCTAGCGTGGTGGATGTGGGTGA
CTTCACAATATCAATCAACGAGGGTCTCCCCTCTGGGGTGCC
CTGCACCTCCCAATGGAACTCCATCGCCCACTGGCTTCTCA
CTCTCTGTGCGCTCTCTGAAGTTACAAATCTGTCCCCTGACA
TCATACAGGCTAATTCCCTCTTCTCCTTCTATGG

- Pattern matching approach:
  - Compare sequence patterns (substrings, subsequences, other features)
  - Compute **similarity** based on pattern comparison

Similarity score: $K(X,Y) = \langle F(X), F(Y) \rangle$

String kernel

Fixed-length vector of features derived from sequence

String Kernel Concept

X

GGAATTTGAGCAGGACTAATTGGAACTTCTTTAAGATTACTTATTCGAACTGAATTAGGAACCCCAGGATCTTTAATTGGAGATGATCAAATTTATAATACAAT
TGTTACAGCTCATGCATTTATTATAATTTTTTTTATAGTTATACCTATTATAATCGGAGGATTTGGAAATTGACTAGTTCCATTAATAATAGGTGCCCCAGATATAG
CTTTCCCCCCGTATAAATAACATAAGATTTTGATTATTACCCCCATCTTTAACTTTATTAATTTCAAGAAGAATTGTTGAAAATGGGGCTGGTACAGGATGAACA
GTTTATCCCCCTCTTTCATCAAATATCGCCCATCAAGGAGCATCTGTTGATTTAGCAATTTTTTCCCTTCATCTTGCTGGTATTTCATCAATTCTTGGAGCTA
TTAATTTTATTACAACAATTATTAATATACGAATTAATAATTTATCTTTTGATCAAATACCATTATTTGTTTGAGCTGTAGGAATTACAGCATTATTATTATTACTTTC
ATTACCTGTTTTAGCAGGTGCTATTACTATATTATTAACAGATCGAAATTTAAATACTTCTTTTTTTGATCCTGCAGGAGGAGGAGATCCAATCTTATACCAACA
CTTATTT

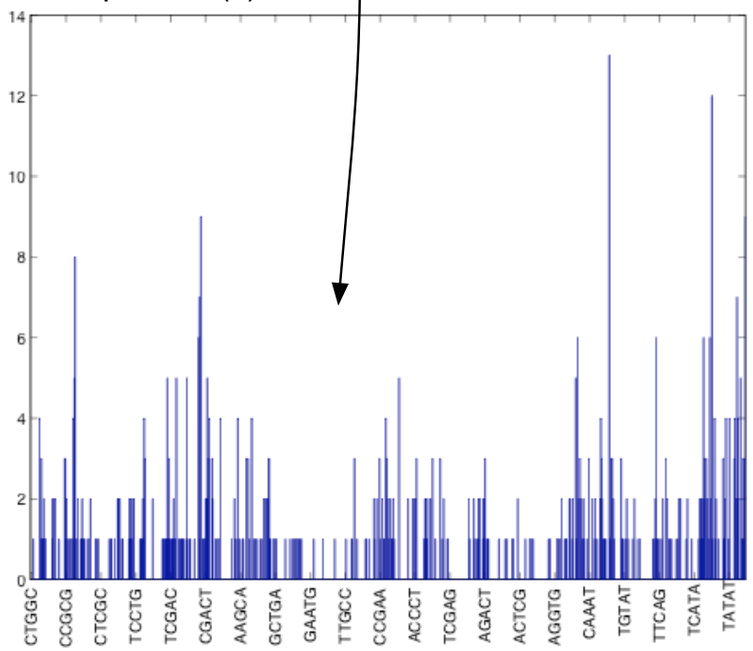
GGAAT

Length k=5

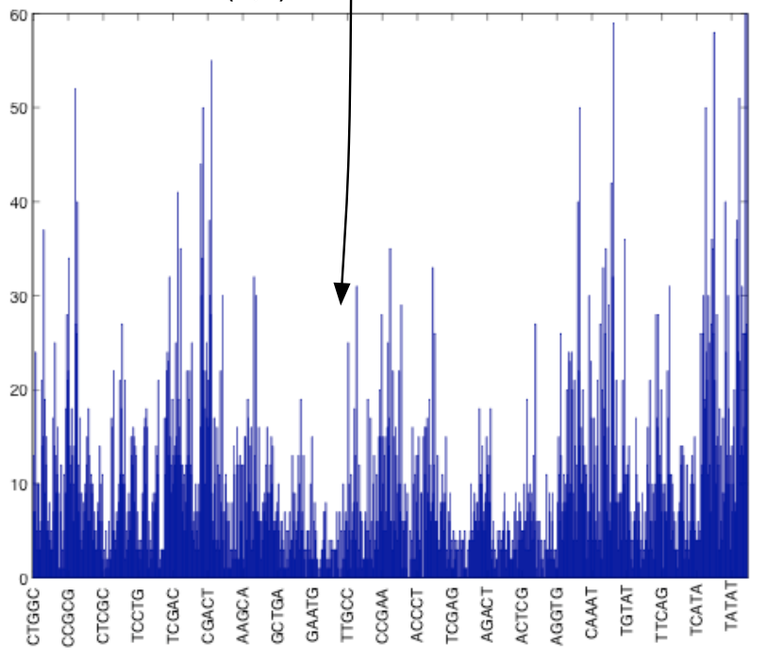
xGAAT
GxAAT
GGxAT
GGAxT
GGAAx

Up to m=1 mismatch

Spectrum(5)



Mismatch(5,1)



F(X)

Similar spectra ~ High Similarity

String Kernel Concept

x

GGAATTTGAGCAGGACTAATTGGAACTTCTTTAAGATTACTTATTCGAACTGAATTAGGAACCCCAGGATCTTTAATTGGAGATGATCAAATTTATAATACAAT
TGTTACAGCTCATGCATTTATTATAATTTTTTTTATAGTTATACCTATTATAATCGGAGGATTTGGAAATTGACTAGTTCCATTAATAATAGGTGCCCCAGATATAG
CTTTCCCCCCGTATAAATAACATAAGATTTTGATTATTACCCCCATCTTTAACTTTATTAATTTCAAGAAGAATTGTTGAAAATGGGGCTGGTACAGGATGAACA
GTTTATCCCCCTCTTTCATCAAATATCGCCCATCAAGGAGCATCTGTTGATTTAGCAATTTTTTCCCTTCATCTTGCTGGTATTTCATCAATTCTTGGAGCTA
TTAATTTTATTACAACAATTATTAATATACGAATTAATAATTTATCTTTTGATCAAATACCATTATTTGTTTGAGCTGTAGGAATTACAGCATTATTATTATTACTTTC
ATTACCTGTTTTAGCAGGTGCTATTACTATATTATTAACAGATCGAAATTTAAATACTTCTTTTTTTGATCCTGCAGGAGGAGGAGATCCAATCTTATACCAACA
CTTATTT

GGAAT

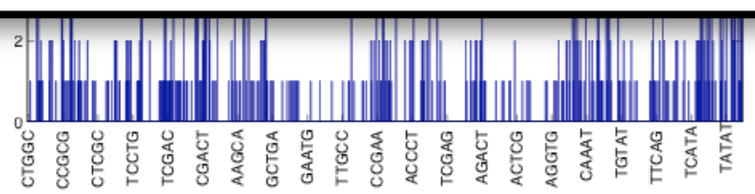
Length k=5

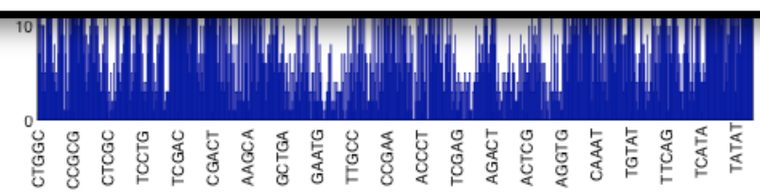
xGAAT
GxAAT
GGxAT
GGAxT

Up to m=1 mismatch

Two important questions:

1. What are the features (representation)?
2. How efficient is the similarity evaluation? (quadratic? linear?)





r(x)

Similar spectra ~ High Similarity

String Kernels

- Kernel functions *K(x,y)* on sequences from alphabet ∑
  - Pairwise-alignment algorithms (Needleman-Wunsch)
    Not Mercer kernels [Vert et al.'04]
  - Pair HMMs [Watkins'99], convolution kernels [Haussler'99], gappy n-gram kernels [Lodhi et al.'02], rational kernels [Cortes et al.'02]

    O(n²) complexity in sequence length ***n*** / pair
  - *Spectrum kernels* [Leslie'02], *mismatch kernels* [Leslie'04], substring kernels [Vishwanathan & Smola'02]

    O(n) complexity in sequence length ***n*** / pair

- *Accuracy & Algorithmic complexity still insufficient for large-scale sequence comparison / annotation*

Spectrum/Mismatch Kernels (I)

- Feature map ($|\Sigma|^k$-dim. representation) based on counts of k-length substrings

$$\Phi_{k,m}(\gamma|X) = \left(\sum_{\alpha \in X} I_m(\alpha, \gamma) \right)_{\gamma \in \Sigma^k}$$

substring length *k*

sequence

feature count for γ

$$I_m(\alpha, \gamma) = \begin{cases} 1, d(\alpha, \gamma) \leq m \\ 0, \text{otherwise} \end{cases}$$

indicator (matching) function

Hamming distance

Example: k-mer feature space for DNA ($|\Sigma|$=4)

AA	AC	AG	...	TT
1	2	3	...	$\|\Sigma\|^k$

feature count

Very important in practice!

exact (m=0) or approximate/smoothed (m>0)

Spectrum kernels (2)

- Spectrum kernel function K(X,Y)

  cumulative pairwise comparison of all substrings *a* and *b* contained in X and Y

$$K(X,Y|k,m) = \sum_{\gamma\in\Sigma^k} \Phi_{k,m}(\gamma|X)\Phi_{k,m}(\gamma|Y).$$

$$= \sum_{\alpha\in X}\sum_{\beta\in Y}\sum_{\gamma\in\Sigma^k} I_m(\alpha,\gamma)I_m(\beta,\gamma)$$

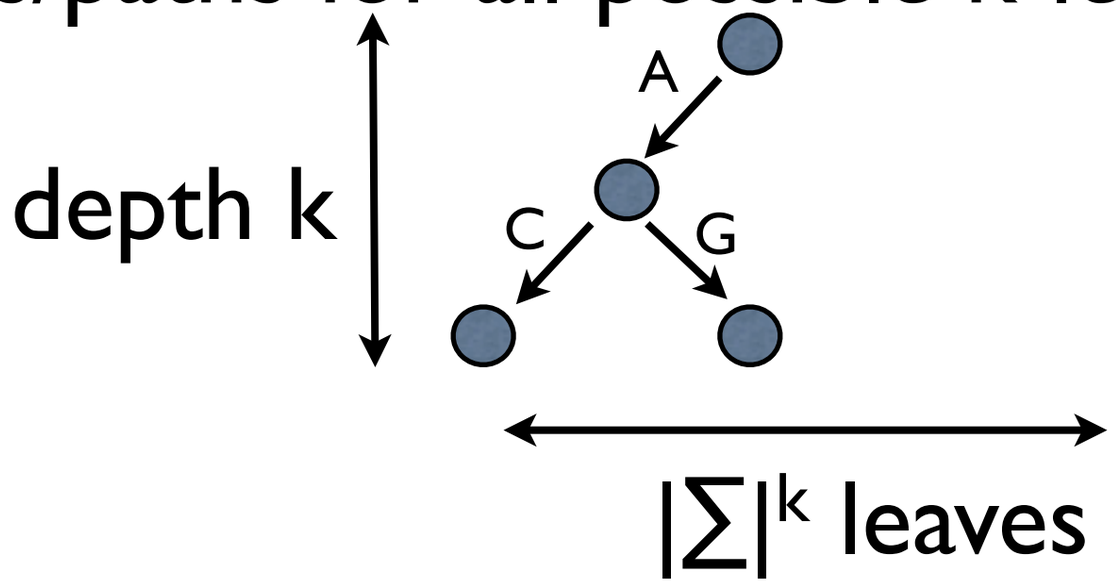
number of similar substrings shared by(a,b)

What are the best algorithms for string kernels?

- Suffix trees (Vishwanathan & Smola, 2002)
  - linear-time all-substring kernel
- (Sparse) dynamic programming (Rousu, 2005)
  - gapped kernel
- ***Mismatch trie*** (Leslie, 2004)
  - applies to mismatch, spectrum, gapped kernels, etc (*inexact matching!!!*)
  - Complexity $O(k^{m+1}|\Sigma|^m(|X| + |Y|))$
    - limits applicability to small k,m,|∑|

Mismatch Trie Algorithm

- Depth-first search over complete tree with leaves/paths for all possible k-length substrings



- each node has a list of substrings that match node's prefix
- branches correspond to all discrete alphabet symbols

- Applies to spectrum, mismatch, gapped, etc kernel computation $O(k^{m+1}|\Sigma|^m(|X| + |Y|))$

- Problematic for large-Σ inputs and relaxed matching (larger m)

Want to address this

Our Results

- New *sufficient-statistic* based algorithms for kernel computation
  - Improve *complexity* bounds over existing algorithms, *removes dependency on the alphabet size*

$$O(k^m|\Sigma|^m n) \Rightarrow O(c_{k,m} n)$$

- Several *orders-of-magnitude* speed-up ⇒

  can now be applied to large-alphabet inputs (music, text, time series, etc)

- Enhance performance: more complex kernels improve predictive accuracy

Our approach: Sufficient Statistics

Evaluation of string kernel functions using *Sufficient Statistics*

Two steps

- (I) Exact spectrum computation with counting sort
- (2) *Sufficient statistics* for string kernels and their computation using (1) as sub-algorithm

(I) Exact Spectrum computation with counting sort

Input sequences

X1: GGAA

X2: TTTGAA

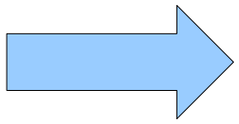
X3: GGAAT

substring kernel

k=3,m=0

Features:	Seq. Index:
GGA	1
GAA	1
TTT	2
TTG	2
TGA	2
GAA	2
GGA	3
GAA	3
AAT	3

Counting sort



AAT	3	Seq. index: 3, Counts: 1
GAA	1	Seq. index: 1 2 3, Counts: 1 1 1
GAA	2	
GAA	3	
GGA	1	Seq. index: 1 3, Counts: 1 1
GGA	3	
TTA	2	...
TTG	2	
TTT	2	Seq. index: 2, Counts: 1

Counting sort:
O(kn) evaluation for exact spectrum

Kernel updates (per *substring*):

$K(J,J) = K(J,J) + cc^T$

c - feature counts, J - sequence index

How does this extend to *inexact matching* (m>0)?

Mismatch Kernel

Main issue: *denser* substring spectrum, many more features due to approximate counts (m>0)

I: $$K(X,Y|k,m) = \sum_{\gamma\in\Sigma^k} \Phi_{k,m}(\gamma|X)\Phi_{k,m}(\gamma|Y).$$

$$= \sum_{\alpha\in X}\sum_{\beta\in Y}\sum_{\gamma\in\Sigma^k} I_m(\alpha,\gamma)I_m(\beta,\gamma)$$

$$I_m(\alpha,\gamma) = \begin{cases} 1, d(\alpha,\gamma)\leq m \\ 0, \text{otherwise} \end{cases}$$

introduces for every *a* mismatch neighborhood *N*(*a*,*m*) of size $O(k^m|\Sigma|^m)$

II: $$K(X,Y|k,m) = \sum_{i_x=1}^{n_x-k+1}\sum_{i_y=1}^{n_y-k+1}\sum_{a\in\Sigma^k} I_m(a, x_{i_x:i_x+k-1})I_m(a, y_{i_y:i_y+k-1})$$

$$= \sum_{i_x=1}^{n_x-k+1}\sum_{i_y=1}^{n_y-k+1} |(N(x_{i_x:i_x+k-1},m)\cap N(y_{i_y:i_y+k-1},m)|$$

$$= \sum_{i_x=1}^{n_x-k+1}\sum_{i_y=1}^{n_y-k+1} \mathcal{I}(x_{i_x:i_x+k-1}, y_{i_y:i_y+k-1})$$

intersection size for two mismatch neighborhoods

Observation I: number of substrings within distance *m* from both *a* and *b* is *independent* of *a* and *b*

I: Intersection Algorithm

Algorithm 1. (Hamming-Mismatch) Mismatch algorithm based on Hamming distance

Input: strings $X, Y, |X| = n_x, |Y| = n_y$, parameters k, m, lookup table $\mathcal{I}$ for intersection sizes
Evaluate kernel using Equation 5:
$K(X, Y|k, m) = \sum_{i_x=1}^{n_x-k+1} \sum_{i_y=1}^{n_y-k+1} \mathcal{I}(d(x_{i_x:i_x+k-1}, y_{i_y:i_y+k-1})|k, m)$
where $\mathcal{I}(d)$ is the intersection size for distance d
Output: Mismatch kernel value $K(X, Y|k, m)$

- *Independent* of alphabet size ∑ and mismatch degree *m*
- *In-place* (only uses $\min(2m,k)+1$ extra space for an auxiliary look-up table)
- Only linear in k (not k^m)
- Quadratic O(|X||Y|) in sequence length
- Can be used with other kernels given corresponding lookup table

Sufficient Statistics for String Kernels

Observation II: kernel is incremented only by min(2m,k)+1 distinct values

$$K(X,Y|m,k) = \sum_{i_x=1}^{n_x-k+1} \sum_{i_y=1}^{n_y-k+1} \mathcal{I}(x_{i_x:i_x+k-1}, y_{i_y:i_y+k-1}) = \sum_{i=0}^{\min(2m,k)} M_i \mathcal{I}_i$$

Matching (sufficient) statistics:
M_i = number of pairs of substrings (a,b) at Hamming distance d(a,b)=i

How to compute matching statistics M_i efficiently?

- Problem: direct computation of M_i is still *quadratic*!

Auxiliary problem

Mismatch Statistic Counting: Given a set of n k-mers from two strings X and Y, for each Hamming distance $i = 0, 1, ..., \min(2m, k)$, output the number of k-mer pairs $(a, b), a \in X, b \in Y$ with $d(a, b) = i$.

- Efficient direct computation of the *number* of substring pairs (a,b) at distance *i* (M_i) is difficult (requires *quadratic* time!)
- Avoiding *quadratic* time:
  - compute *inexact*(!) statistics C_i first (in *linear* time)
  - then obtain exact matching statistics M_i in *closed form* from the set of inexact statistics

Computing matching statistics

- Instead of Mi first compute *approximate* (with over-counting) number of pairs C_i at distance *at most i*

- *Algorithm*: iteratively remove *i* positions, sort and compute spectrum kernel for (*k-i*)-length substrings

$$d(a', b') = 0 \Rightarrow d(a, b) \leq i$$

- Inexact matching statistics C_i:

$$C_i = M_i + \Sigma_{j=0}^{j=i-1} \binom{k-j}{i-j} M_j$$

II: Sufficient Statistic (SS) algorithm

Algorithm. **(Mismatch-SS) Mismatch kernel algorithm based on Sufficient Statistics**
Input: strings X, Y, parameters k,m, $\mathfrak{I}_i$
1. Compute min(2m,k) *cumulative* matching statistics, C_i, using counting sort
2. Compute *exact* matching statistics, M_i

$$M_i = C_i - \Sigma_{j=0}^{i-1} \binom{k-j}{i-j} M_j, \quad i = 0,\ldots, min(min(2\mathrm{m}, k), k-1)$$

$$M_k = T - \Sigma_{j=0} k-1 M_j$$

3. Evaluate kernel using :

$$K(X, Y \mid m, k) = \Sigma_{i=0}^{min(2\mathrm{m},k)} M_i \mathfrak{I}_i$$

Output: Mismatch kernel value K(X,Y|k,m)

$$O(k^{m+1} |\Sigma|^m n) \quad \Rightarrow \quad O(c_{k,m} n), \; c_{k,m} = \Sigma_{i=0}^{2\mathrm{m}} \binom{k}{i} (k-i)$$

original complexity (trie-based)

Sufficient-statistic based

Using Sufficient Statistics to compute String Kernel Functions

- Spectrum kernels (Leslie, 2002)

$$K(X,Y) = M_0$$

- Gapped kernels (Leslie, 2004; Rousu, 2005)

$$K(X,Y) = \Sigma_{i=0}^{i=m} C_i'$$

- Spatial kernels (Kuksa, 2008)

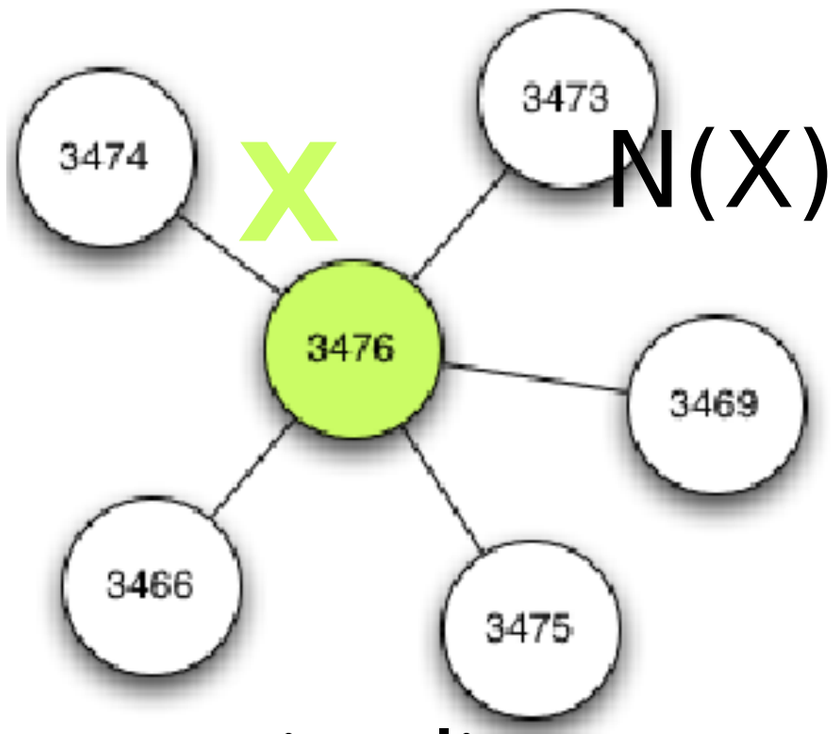
$$K(X,Y) = \Sigma_{\boldsymbol{d}=(d_1, d_2, .., d_{t-1})} C_{\boldsymbol{d}}$$

- Reduce computation to multiple rounds of exact spectrum kernel computation (counting sort)
- Extends to *semi-supervised settings: (I) cluster kernels (Weston, 2005)*, (II) abstraction kernels (*Kuksa, 2010*)

Neighborhood Kernels

$$K(X,Y) = \sum_{x \in N(X)} \sum_{y \in N(Y)} K(x,y)$$

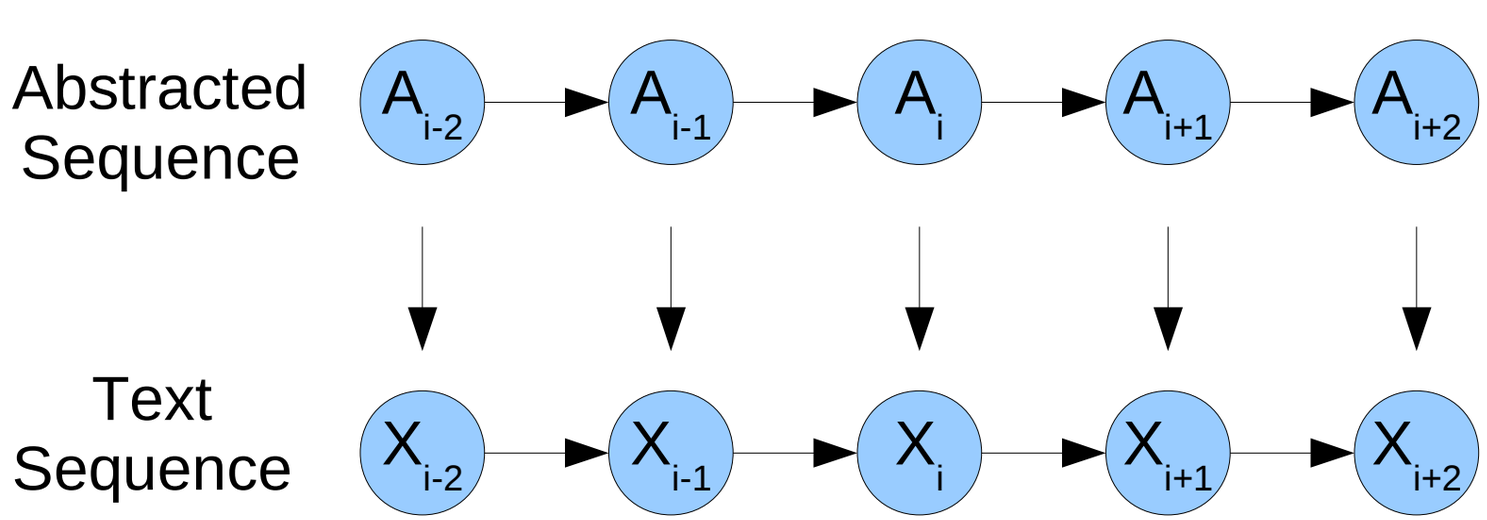
(Weston, 2005)



- N(X) - set of sequences neighboring X
- Can be labeled or unlabeled
  (semi-supervised learning)
- Direct evaluation is quadratic!
- Can extend our approach and make computation linear
  - Jointly sort N(X), N(Y)
  - Apply desired similarity measure (mismatch, spectrum, gapped, etc)
  - Works with millions of sequences

Abstraction-augmented kernels

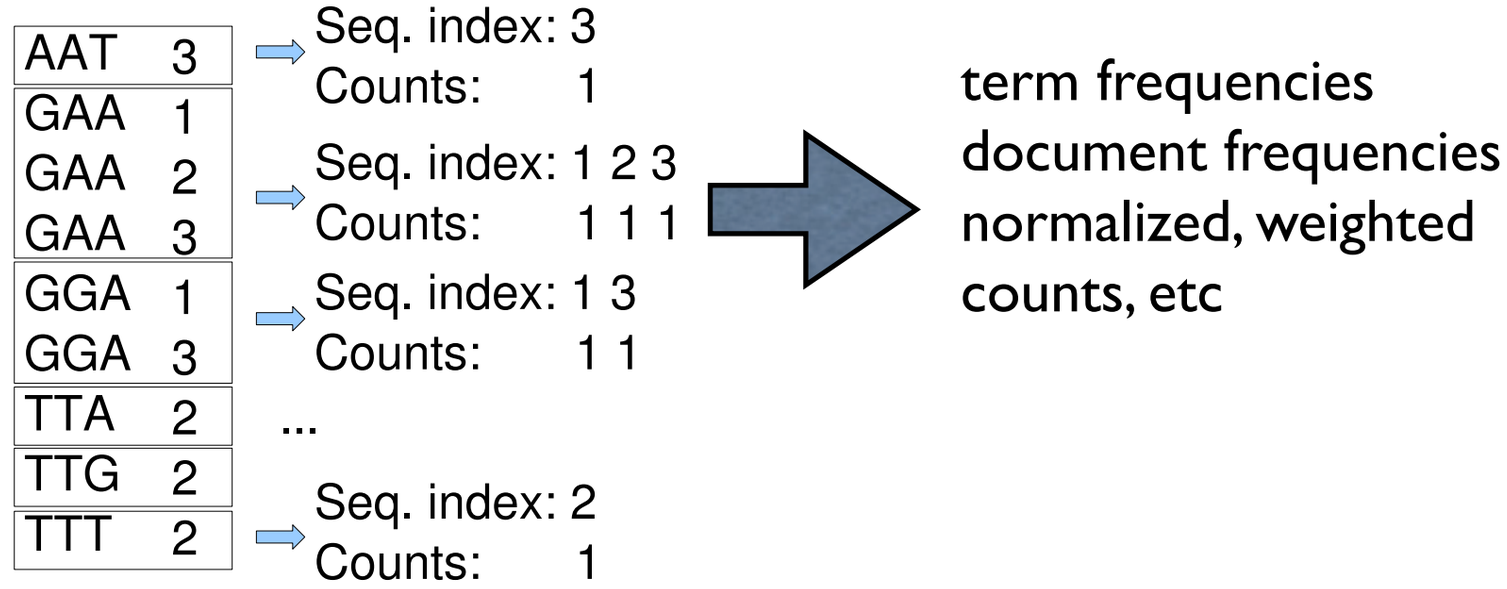
- Learn word abstractions using unsupervised (or semi-supervised) embedding and clustering (ECML 2010)



- Apply kernel to both original sequence X

Weighted Kernels

- Can use with weighted embeddings
  - Multinomial Manifold Kernels (Zhang, SIGIR 2005), geodesic distance kernels
  - TF-IDF embedding, L1, L2 normalization
  - Gap-weighted kernels, etc



Summary

- New family of algorithms based on *sufficient statistics* and counting sort for string kernels
  - unified approach
  - improves complexity bounds over existing algorithms
  - reduces computation to exact spectrum kernel computation

Evaluation I:

time efficiency

Running time (I)

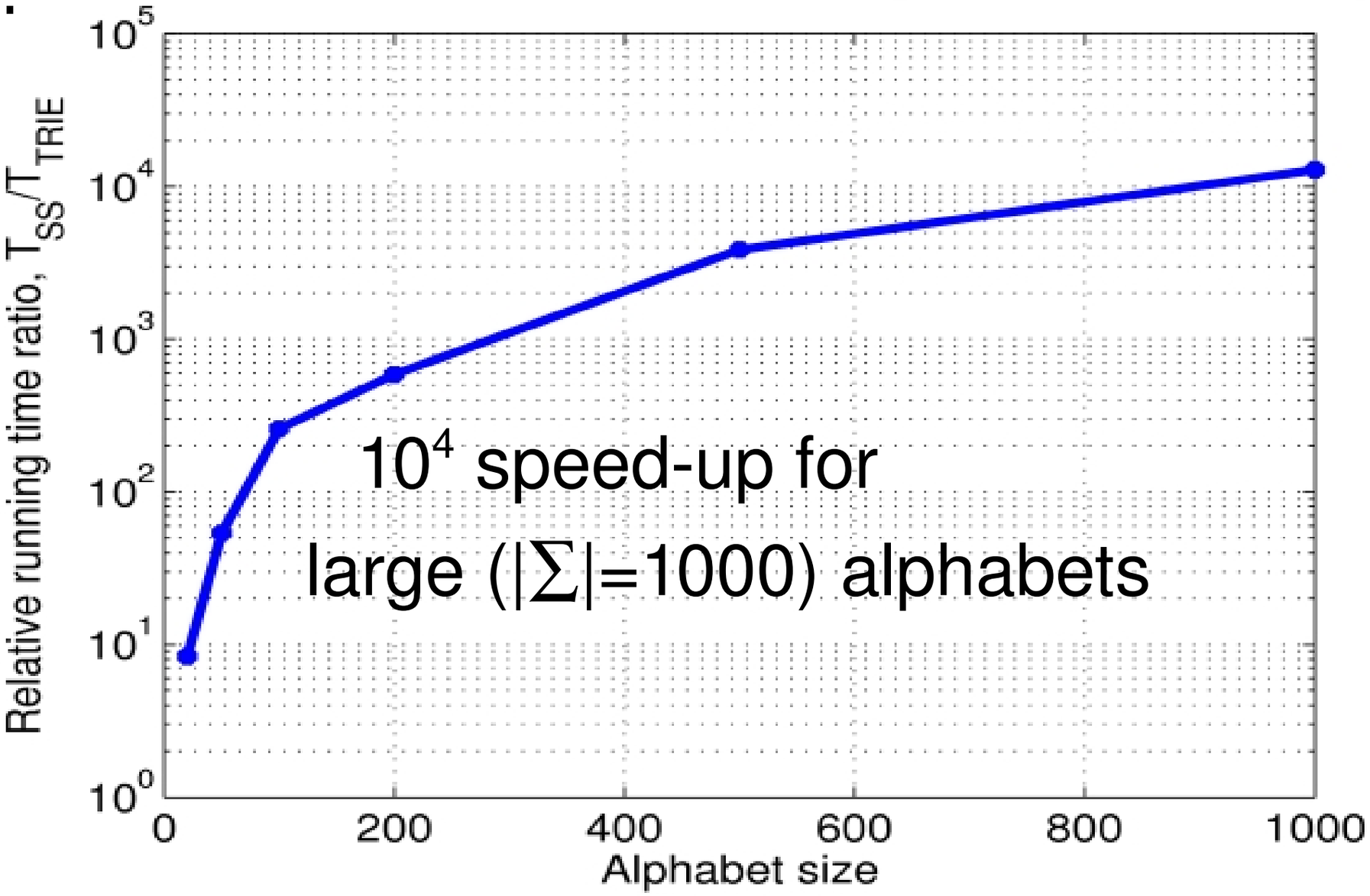


Figure 1: Relative running time vs. alphabet size

Settings: seq. length = 10K, k=5, m=1

Speed-up

Protein	100
Music	~10^4
Text	~10^6

Running time (2)

Kernel computation K(X,Y), real data

Running time, (s)

	long protein	protein	dna	text	music
n	36672	116	570	242	6892
$\|\Sigma\|$	20	20	4	29224	1024
(5,1)-trie	1.6268	0.0212	0.0260	20398	526.8
(5,1)-ss	0.1987	0.0052	0.0054	0.0178	0.0331
time ratio	8	4	5	10^6	16,000
(5,2)-trie	31.5519	0.2918	0.4800	-	-
(5,2)-ss	0.2957	0.0067	0.0064	0.0649	0.0941
time ratio	100	44	75	-	-

Evaluation II:

- Text
- Relation extraction
- Music
- Bio-informatics:
  - DNA Barcoding
  - Protein structural annotation

Relevant Document Detection

- BioCreativeII competition data: detect PPI relevant documents from their abstracts
  - ~5K abstracts, ~1.2M words
  - unlabeled: 1.4M abstracts, 1.3G words

Method	**F1**
Semi-supervised Abstraction Kernel (Kuksa et al, 2010)	80.11
Baseline I: BioCreativeII competition (best)	78.00
Baseline II: TF-IDF	73.98

Relation Extraction

Example:

The protein product of **c-cbl** proto-oncogene is known to interact with several proteins, including **Grb2**, **Crk**, and **PI3 kinase**, and to regulate signaling …

Interacting pairs: (c-cbl, Grb2), (c-cbl, Crk), etc.

- PPI Relation extraction (sentence-level), ~4K sentences, F1 scores (AIMED dataset)

Method	**F1**
Mismatch	**64.48**
Baseline I: Multiple kernel, multiple parser (Makoto et al, 2008)	**61.4**
Baseline II: Dependency and deep parsers (Miyao et all, 2008)	**59.5**

Text classification

- Reuters, test F1 scores, |∑|=29,224

Comparison with state-of-the-art

Class	TF-IDF	KSG	Double	SS-4†	NG-4‡
Acq	97.11	96.8	**97.72**	88.0	93.20
Crude	87.71	89.4	**90.11**	84.0	84.80
Earn	98.70	98.3	**99.02**	97.0	98.40
Grain	**93.29**	92.5	90.18	84.0	84.0
Interest	71.80	81.5	**81.63**	66.0	71.90
Money	77.61	**84.0**	82.66	76.0	75.70
Ship	72.97	81.9	**85.56**	65.0	62.60
Trade	84.26	90.2	**93.47**	73.0	77.90
Mean (macro-average)	85.43	89.33	**90.05**	79.13	81.06
Mean (micro-average)	93.18	93.9	**94.51**	-	-

SS = subsequence kernel
NG = n-gram kernel
KSG = key-substring-group method
TF-IDF = tf-idf kernel

Music Genre Prediction

- 10 genres, ~30s of audio per song
- VQ with $|\Sigma|=1024$ codewords (MFCC feat.), seq. length ~7000

Method	**Error**
Mismatch kernel	35.6
Spatial kernel	29.4
Baseline 1: DWCH (Daubechies Wavelet), Li et al	41.6

Music Artist Recognition

- Album-wise cross-validation over 20 artists (120 albums)
- VQ with $|\Sigma|=1024$ codewords

Method	**Error**
Mismatch kernel	44.66
Spatial kernel	32.50
Baseline 1: GMM	44.0

DNA Barcoding (1)

Dataset	# species	# barcodes
ACG	573	4267
Hesperiidae	364	2185
Astraptes	12	465
Bats of Guyana	96	840
Birds of North America	656	2589
Fish of Australia	211	754
Fish larvae	7	35

- Task: assign DNA barcode to its species group
- Large number of classes (species), limited labeled data

DNA Barcoding (2)

cross-validation error:

	Mismatch	Hamming	Kimura	Smith-Waterman
ACG	**2.37±0.81**	11.44 ± 1.52	5.51 ± 0.86	3.66 ± 0.66
Hesperiidae	**3.48±1.02**	14.49 ± 2.36	3.81 ± 1.26	5.45 ± 1.20
Astraptes	**1.07±1.81**	3.61 ± 2.77	1.71 ± 1.96	1.64 ± 1.03
Bats Guyana	**1.63±1.22**	2.72 ± 1.83	**1.63 ± 1.22**	**1.63 ± 1.22**
Birds of North America	**6.10±1.52**	18.38 ± 2.05	**6.02 ± 1.36**	8.20 ± 1.53
Fish Australia	**5.35±3.36**	5.87 ± 4.01	**5.35 ± 3.36**	5.35 ± 3.36
Fish larvae†	**2.86**	11.43	8.57	5.71

Linear time

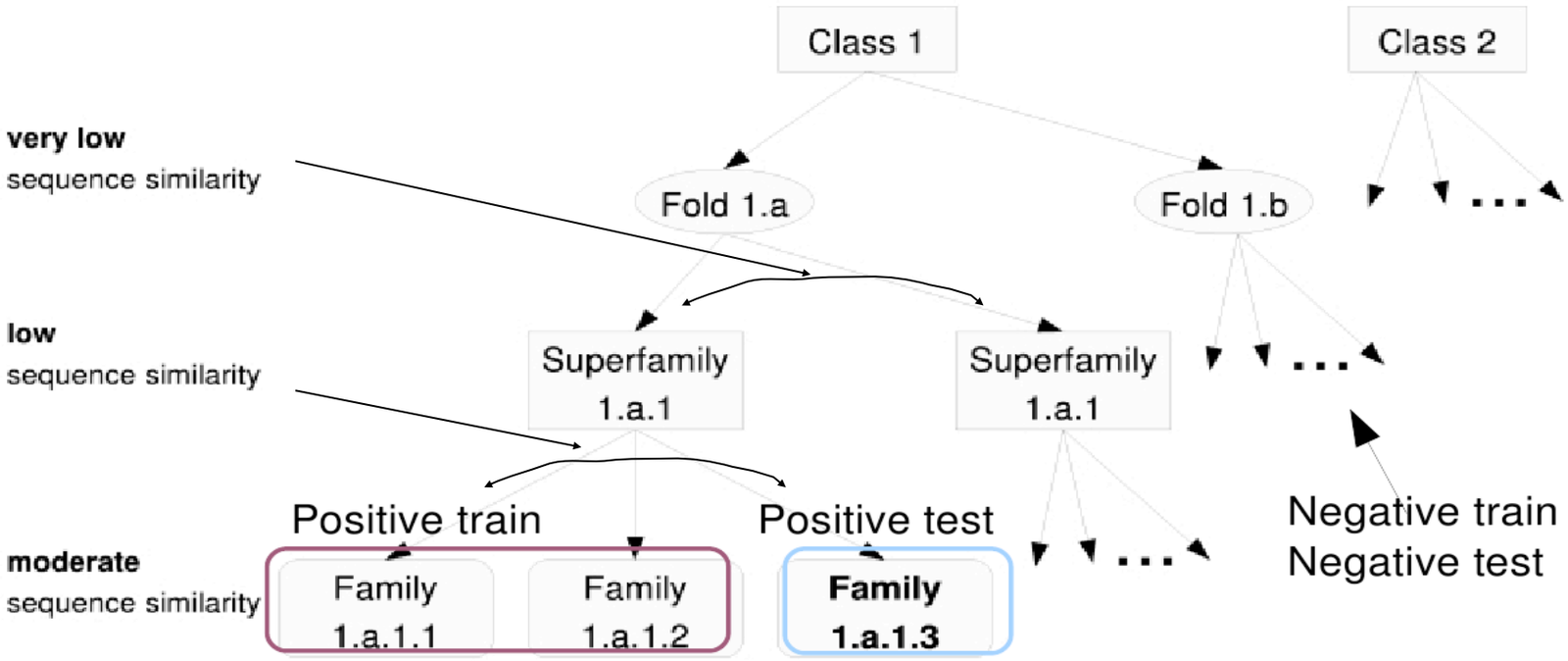
Quadratic

Large-scale protein annotation

- Semi-supervised setting:
  - 2862 labeled sequences (SCOP)
  - 1M unlabeled sequences (NR dataset)

Method	Multi-class fold prediction (27 classes)	Remote homology prediction (54 superfam)
SSSK (triple)	**77.62**	**89.44**
State-of-the-art: profile kernel	67.83	81.51

Structural classification of proteins (SCOP)



Summary

- New string kernel methods and algorithms
- **Benefit I** (complexity): fast, alphabet-free sequence matching based on sufficient statistics
- **Benefit II** (accuracy): enhances performance on many practical tasks (text, music, bio-informatics)
- Extends to richer semi-supervised settings
  - neighborhood kernels
  - abstraction-based kernels
- Outlook
  - Sequence modeling challenge: continuous and no-word-boundary sequences
  - Similarity search: efficient indexing & retrieval algorithms
  - Web applications

Thank you!

References

- Bio-Informatics
  - Pavel Kuksa, Pai-Hsi Huang, and Vladimir Pavlovic. Efficient use of unlabeled data for protein sequence classification: a comparative study. *BMC Bioinformatics*, 10(Suppl 4):S2, 2009
  - Pavel Kuksa and Vladimir Pavlovic. Efficient alignment-free DNA barcode analytics. *BMC Bioinformatics*, 10(Suppl 14):S9, 2009
  - Pavel Kuksa, Pai-Hsi Huang, and Vladimir Pavlovic. Fast and Accurate Multi-class Protein Fold Recognition with Spatial Sample Kernels. In *Computational Systems Bioinformatics: Proceedings of the CSB2008 Conference*, pp. 133–143, 2008
  - Pavel Kuksa, Pai-Hsi Huang, and Vladimir Pavlovic. Fast Protein Homology and Fold Detection with Sparse Spatial Sample Kernels. In *19th International Conference on Pattern Recognition ICPR 2008*, 2008.
- String kernels
  - Pavel Kuksa, Pai-Hsi Huang, and Vladimir Pavlovic. Scalable Algorithms for String Kernels with Inexact Matching. In *NIPS*, 2008. Spotlight Presentation
  - Pavel Kuksa and Vladimir Pavlovic. Fast Kernel Methods for SVM Sequence Classifiers. In *WABI*, pp. 228–239, 2007.
  - Pavel P. Kuksa and Vladimir Pavlovic. Spatial Representation for Efficient Sequence Classification. In *ICPR*, 2010.
- NLP
  - “Semi-Supervised Abstraction-Augmented String Kernel for Multi-Level Bio-Relation Extraction.”, Pavel Kuksa, Yanjun Qi, Bing Bai, Ronan Collobert, Jason Weston, *ECML*, 2010
  - Pavel Kuksa and Yanjun Qi. Semi-Supervised Bio-Named Entity Recognition with Word-Codebook Learning. In *SDM*, 2010.
  - Yanjun Qi, Pavel P. Kuksa, Ronan Collobert, Kunihiko Sadamasa, Koray Kavukcuoglu, and Jason Weston. Semi-Supervised Sequence Labeling with Self-Learned Features. In *Proc. International Conference on Data Mining (ICDM'09)*, IEEE, 2009.