Machine Learning and Algorithmic Approaches for Biomedical Data Analysis

Pavel P. Kuksa
Machine Learning Department
NEC Laboratories America Inc

This talk: Recent Results

1. Computationally efficient methods for inexact sequence matching
   - generalizes computation and significantly improves time-efficiency of many sequence similarity functions
   - more complex models can be investigated to enhance performance
2. General Similarity Sequence Models
   - go beyond traditional Hamming similarity using more "precise" similarity metrics to improve accuracy
   - retain computational efficiency (linear time) of simpler Hamming methods
3. Large scale semi-supervised method for improving bio-named entity and PPI prediction from biomedical literature

- Part I: Similarity computation under k-mer sequence model

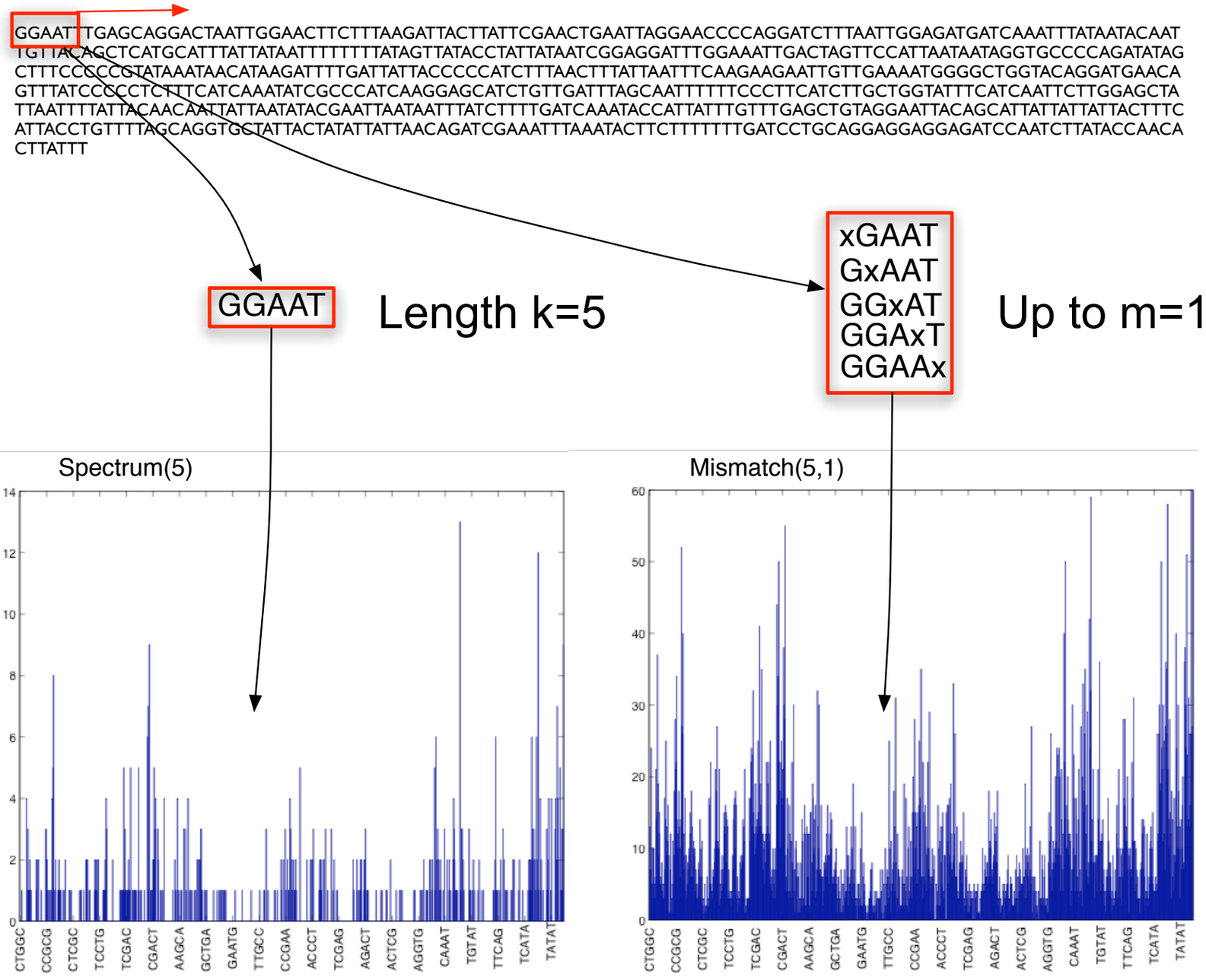
k-mer sequence models

- Used in many bioinformatics applications
  - Genome assembly
  - Multiple sequence alignment
  - Homology search (e.g., BLAST)
  - Sequencing error correction
  - Genomic complexity studies
  - Similarity assessment, classification, clustering

k-mer models and similarity inference

I: Map sequence *X* to fixed-dim feature vector F(*X*)

X



II: Similarity score: K(*X*, *Y*) = <F(*X*), F(*Y*)> ← string kernel

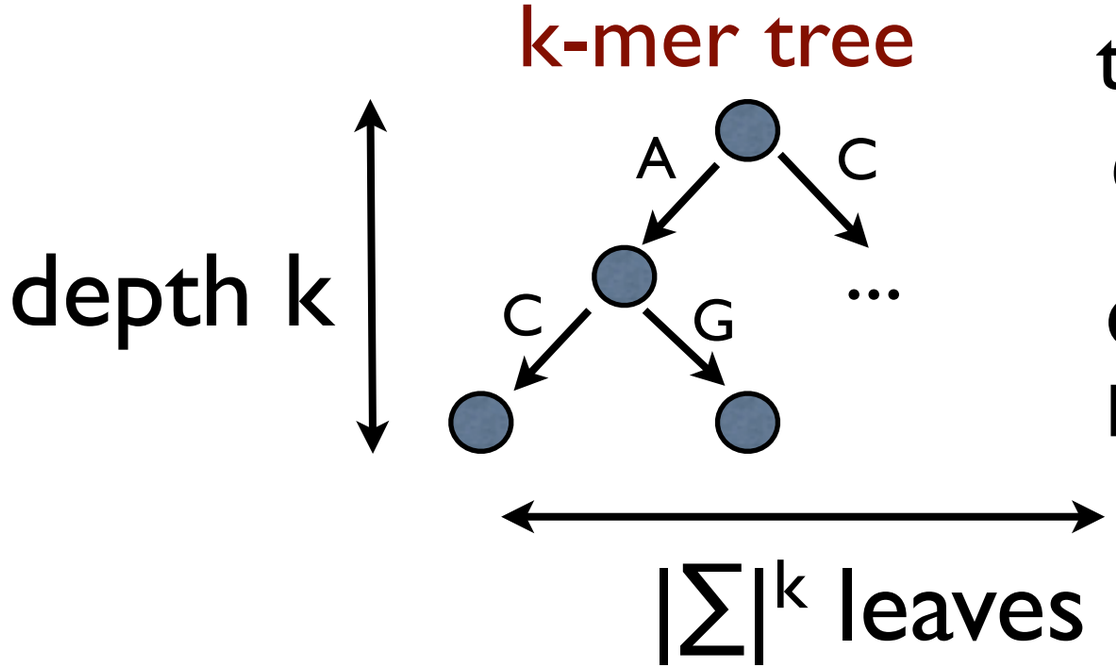
Similar *X* and *Y* ⇒ Similar F(*X*) and F(*Y*) ⇒ large K(*X*,*Y*)

What are the best *k*-mer algorithms?

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

Mismatch Trie Algorithm

- Depth-first search over complete *k*-mer tree with leaves/paths for all possible k-mers



traversal complexity:

$$O(k^{m+1}|\Sigma|^m(|X| + |Y|))$$

$O(k^m |\Sigma|^m)$ - number of substrings at Hamming distance of at most *m*

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

Want to address this

Our Results

- New *sufficient-statistic* based algorithms for similarity/ kernel computation with inexact k-mer matching
  - Improve *time complexity* over existing algorithms, *removes dependency on the cardinality of the alphabet*

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

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

  can now be applied to larger inputs, longer substrings *k*, with many mismatches *m*

- *Enhance performance*: more complex kernels improve predictive accuracy

Our approach: Sufficient Statistics

Evaluation of sequence similarity in k-mer models using *Sufficient Statistics*

Two steps:

(1) Exact spectrum computation with counting sort

(2) *Sufficient statistics* for string kernels and their computation using (1) as a sub-algorithm

(I) Exact *k*-mer spectrum computation

Input sequences

X1: GGAA

X2: TTTGAA

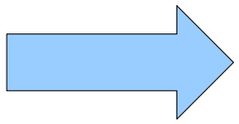
X3: GGAAT

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)?

Approximate sequence matching

Main issue: *denser* k-mer spectrum, i.e. many more features due to approximate matching (m>0)

matching function:

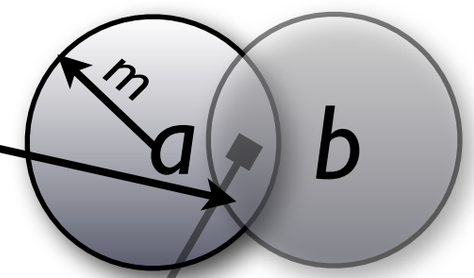
$$I_m(\alpha,\gamma) = \begin{cases} 1, d(\alpha,\gamma) \le m \\ 0, \text{otherwise} \end{cases}$$

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

frequency of γ

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

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



intersection size of the *a* and *b* mismatch neighborhoods

Observation I: number of substrings within distance *m* from both *a* and *b* is *independent* of *a* and *b*; it only depends on *Hamming* distance *d(a,b)*

Sufficient statistics

Observation II: kernel value is incremented only by $\min(2m,k)+1$ distinct values corresponding to Hamming distances 0-2*m* between *k*-mers

$$K(X,Y|k,m) = \underbrace{\sum_{\alpha\in X}\sum_{\beta\in Y}\mathcal{I}(\alpha,\beta)}_{O(|X||Y|)} = \sum_{i=0}^{2m} M_i\mathcal{I}_i$$

Sufficient statistics:

M_i = number of pairs of substrings (*a* in *X*, *b* in *Y*) at Hamming distance $d(a,b)=i$; I_i - intersection size

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

How to compute matching statistics M_i efficiently?

Computing sufficient statistics

- Instead of M_i first compute *approximate* (with *over-counting*) number of pairs C_i at distance *at most i* ($\leq i$) *(*as opposed to *at* distance *i)*

- *Algorithm* (C_i): iteratively remove *i* positions, sort and compute exact *k*-mer spectrum for (*k-i*)-mers *a'*, *b'*

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

- Relationship between cumulative matching statistics C_i and sufficient statistics M_i

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

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 similarity 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 & 2010)

$$K(X,Y) = \Sigma_{\boldsymbol{d}=(d_1, d_{2,}.., d_{t-1})} C_{\boldsymbol{d}}$$

- In all cases reduce computation to multiple rounds of exact spectrum kernel computation (counting sort!)

Summary: algorithms for similarity computation in k-mer models with inexact matching

- New family of algorithms based on *sufficient statistics* and counting sort
  - unified approach
  - improves time complexity and accuracy over state-of-the-art algorithms
  - reduces computation to *exact k-mer spectrum* computation

Empirical evaluation

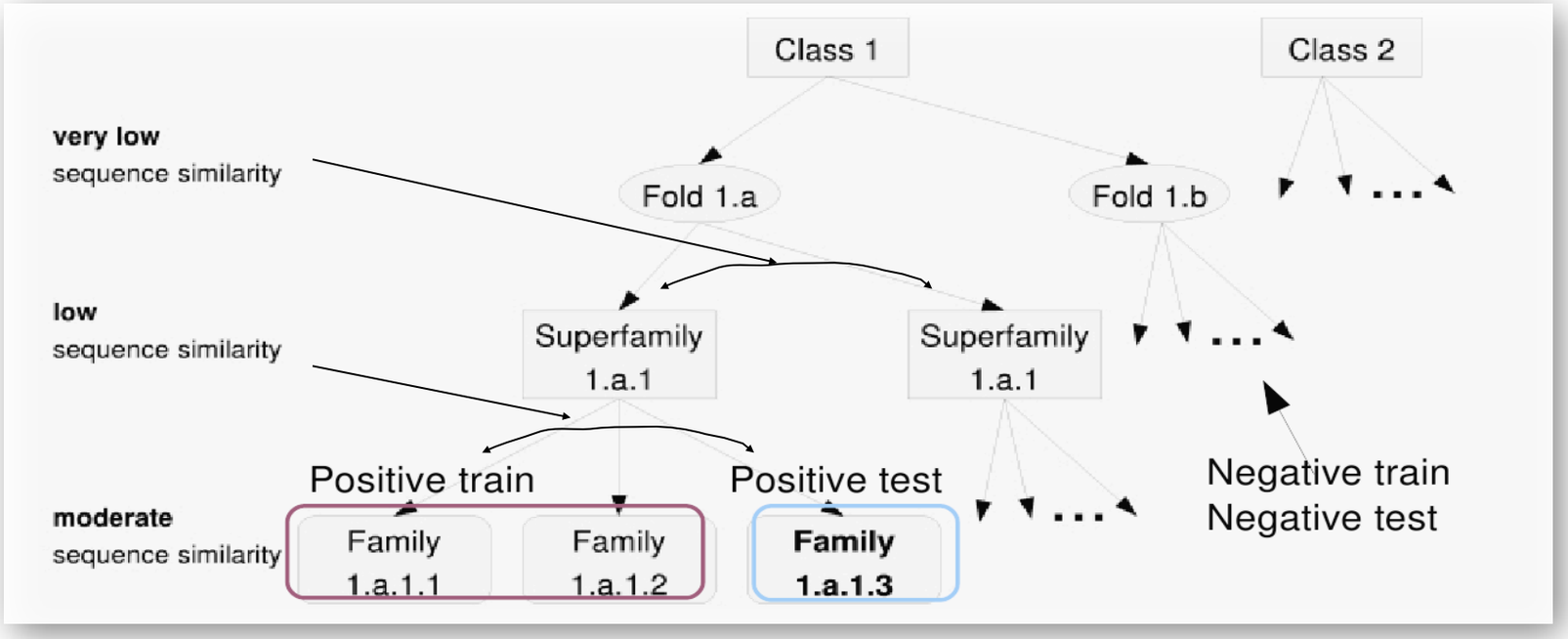
Computational advantage

- Average speed-up in K(X,Y) computation for protein sequences

(k,m)	Speed-up (T_{trie} / T_{ss})
(5,2)	11
(6,2)	72
(7,2)	125
(7,3)	443
(9,3)	834

Protein remote homology prediction

Structural classification of proteins (SCOP)



Ex: prediction task

Target superfamily: 1.a.1		
	positive	negative
train	1.a.1.1 1.a.1.2	other sf/ folds
test	1.a.1.3	other sf/ folds

- Tasks: infer membership of a protein in the target superfamily or fold

- Remote homology prediction: average over 54 superfamily membership problems

Supervised setting

kernel (k,m)	ROC	ROC50
(5,1)	87.75	41.92
(7,3)	91.31	52.00
(9,4)	91.45	53.51
(10,5)	**91.60**	**53.78**

Semi-Supervised setting

kernel	ROC	ROC50
SCOP		
(5,1)	90.93	67.20
(5,2)	91.42	69.35
(7,3)	**92.27**	**73.29**
UniProt		
(5,1)	96.73	81.05
(5,2)	97.05	82.25
(7,3)	**97.78**	**86.32**

- Multi-class protein fold prediction
  - 27 fold classes
  - challenge: very low sequence similarity

kernel (k,m)	Balanced error	F1 score
Supervised		
(5,1)	51.17	61.68
(7,3)	43.60	65.09
Semi-Supervised (NRDB)		
(5,1)	24.02	80.47
(7,3)	**22.49**	**81.04**

Part II:
Generalized Similarity Sequence Models

Traditional k-mer models

- *Feature space*: indexed by all possible *k*-mers: sparse embedding (spectrum, SSSK), dense embedding (mismatch, profile)
- *Hamming distance*-based matching

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

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

$$I(\alpha,\beta) = \begin{cases} 1, h(\alpha,\beta) \leq m \\ 0, otherwise \end{cases}$$

- Drawback: Hamming distance may not reflect true underlying similarity/dissimilarity:
  - ex: different pairs of AA induce different levels of similarity
  - matching of k-mers should reflect biologically meaningful similarity, not only simple character-level differences

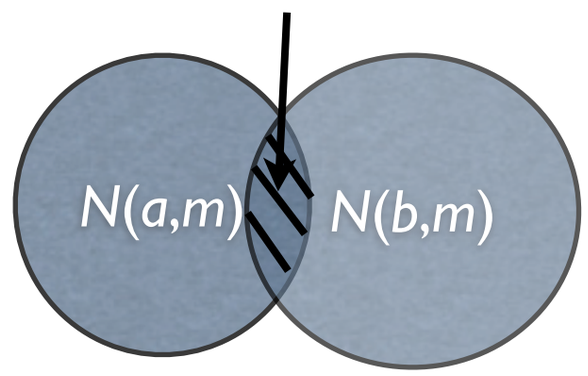
=> need more "precise" similarity measures!

General similarity

- Traditional string kernels: *k*-mer similarity is measured as the number of identical substrings in the mismatch neighborhoods (i.e. intersection size)

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

intersection size



How to define more "precise" and meaningful similarity?

General Similarity functions

- Idea: replace intersection size-based similarity with metric similarity function

$$\sum_{\gamma \in \Sigma^k} I_m(\alpha, \gamma) I_m(\beta, \gamma) \rightarrow \mathcal{S}(\alpha, \beta)$$

- General similarity kernel function

$$K(X, Y | k, \mathcal{S}) = \sum_{\alpha \in X} \sum_{\beta \in Y} \mathcal{S}(\alpha, \beta)$$

  - Complexity in general becomes quadratic!
  - This is not a proper kernel!

=> How to (1) incorporate general metric S and (2) preserve linear time complexity?

Idea: Similarity-preserving embedding

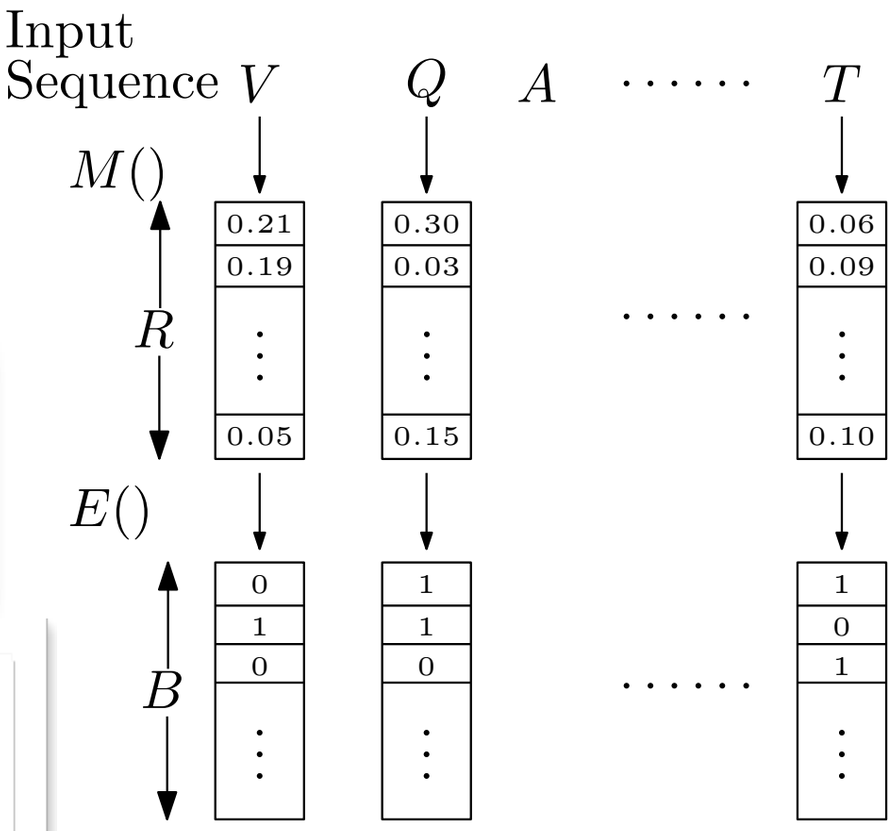
1. Learn binary symbolic embedding *E* for which Hamming distance accurately approximates *S*

$$h(E(a),E(b)) \sim S(a,b)$$

2. Use *Hamming-type computation* to efficiently evaluate kernel $K\ X,Y|\mathcal{S}$ in linear time

$$K(X,Y|k,\mathcal{S}) = \sum_{\alpha \in X} \sum_{\beta \in Y} \mathcal{S}(\alpha,\beta)$$

Learning symbolic embedding



Similarity hashing:

$$\min \sum_{\alpha,\beta} S(\alpha,\beta) h(E(\alpha), E(\beta))^2$$

s.t. $E(\alpha) \in \{-1,1\}^B$

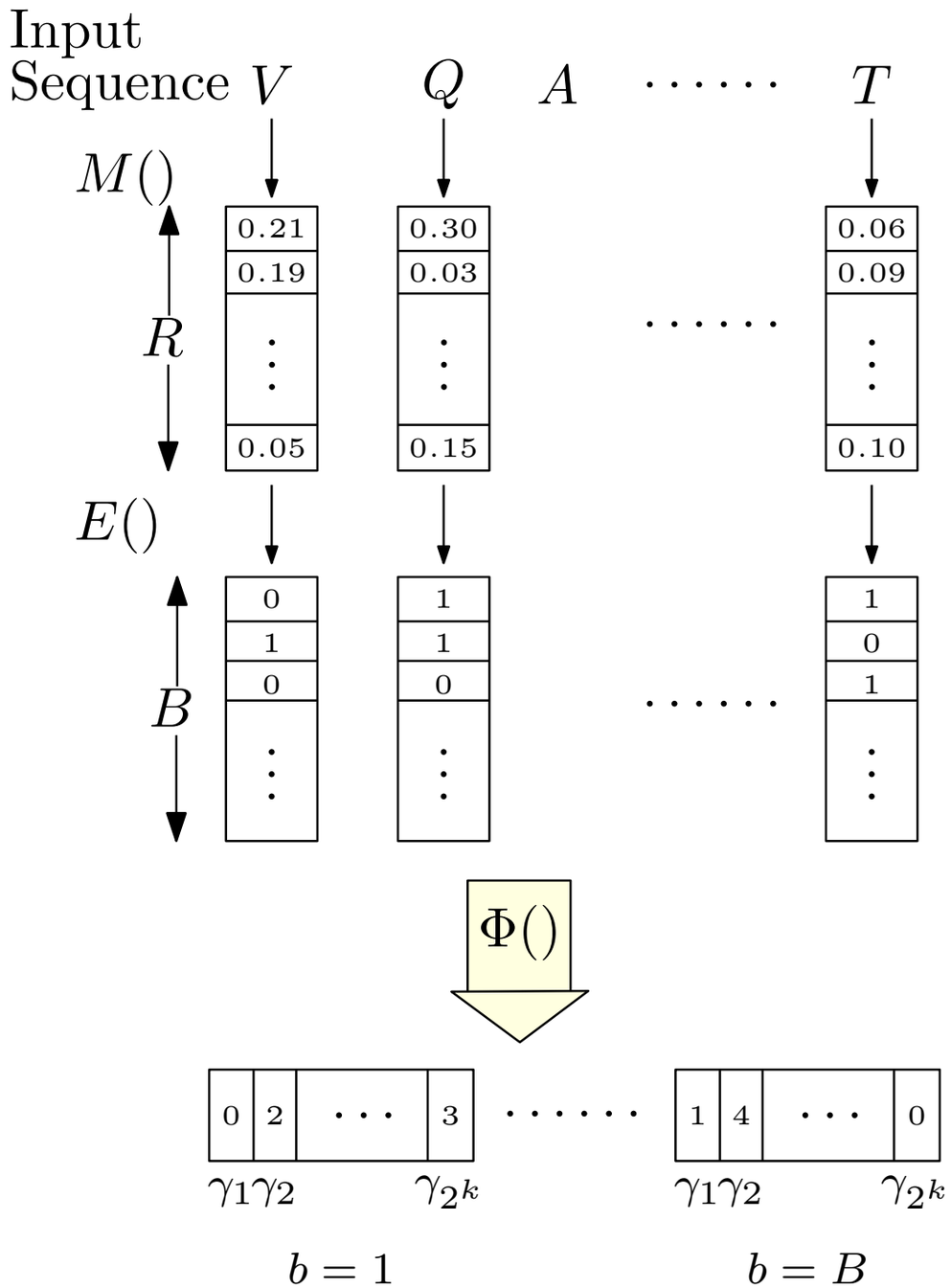
Idea: minimize average Hamming distance *h*(E(*a*),E(*b*)) between data points similar according to S(*a*,*b*)

$$E(X) = E(x_1), ..., E(x_n)$$

$$\mathsf{E}(\mathsf{x}_i) = \mathsf{e}_1^i \mathsf{e}_2^i \ldots \mathsf{e}_B^i \qquad e_j^i \in \{0,1\}$$

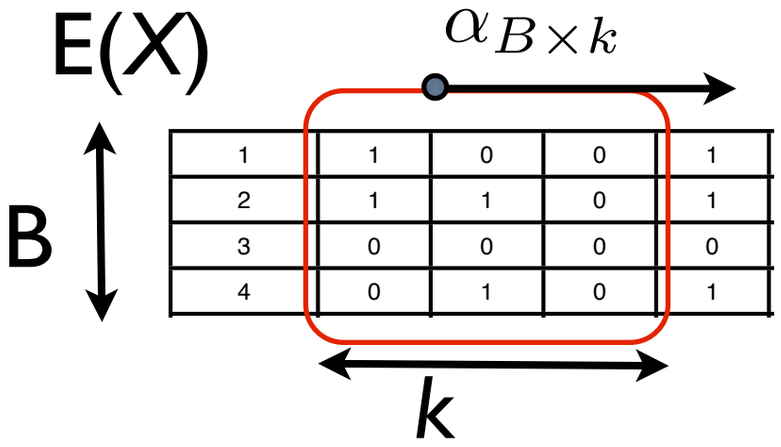
$$h(E(\alpha), E(\beta)) \propto \mathcal{S}(\alpha,\beta)$$

Similarity-preserving embedding



- separate 2^k feature map for each dimension $b=1...B$

Similarity-preserving kernel



$$K_k(E(X), E(Y)) = \langle \Phi_k(E(X)), \Phi_k(E(Y)) \rangle \tag{5.9}$$

$$= \sum_{b=1}^{B} \sum_{\gamma \in \{0,1\}^k} \left(\sum_{\alpha \in E(X)} I(\alpha[b], \gamma) \sum_{\beta \in E(Y)} I(\beta[b], \gamma) \right)$$

$$= \sum_{b=1}^{B} \sum_{\alpha \in E(X)} \sum_{\beta \in E(Y)} I(\alpha[b], \beta[b]) \tag{5.10}$$

$$= \sum_{\alpha \in E(X)} \sum_{\beta \in E(Y)} \underbrace{\sum_{b=1}^{B} I(\alpha[b], \beta[b])}_{s_{\alpha\beta}} \tag{5.11}$$

number similar rows in a and b

Kernel value K(E(*X*),E(*Y*)) approximates general similarity function

$$K(X, Y|k, \mathcal{S}) = \sum_{\alpha \in X} \sum_{\beta \in Y} \mathcal{S}(\alpha, \beta)$$

$$\max\{h_{\alpha\beta} - (k-1)B, 0\} \leq s_{\alpha\beta} \leq \frac{h_{\alpha\beta}}{k}$$

$$h_{\alpha\beta} = h(E(\alpha), E(\beta)) \propto \mathcal{S}(\alpha, \beta)$$

=> K(E(*X*),E(*Y*)) ~ K(*X*,*Y*|S) !

Similarity-preserving kernels

- Benefits:
  - Incorporate metric similarity function S into matching
    - Enhances representation by capturing interrelationships between sequence elements and features according to S
      - =>Refine matching of otherwise symbolically different sequence elements/features
    - Computationally efficient: linear time!

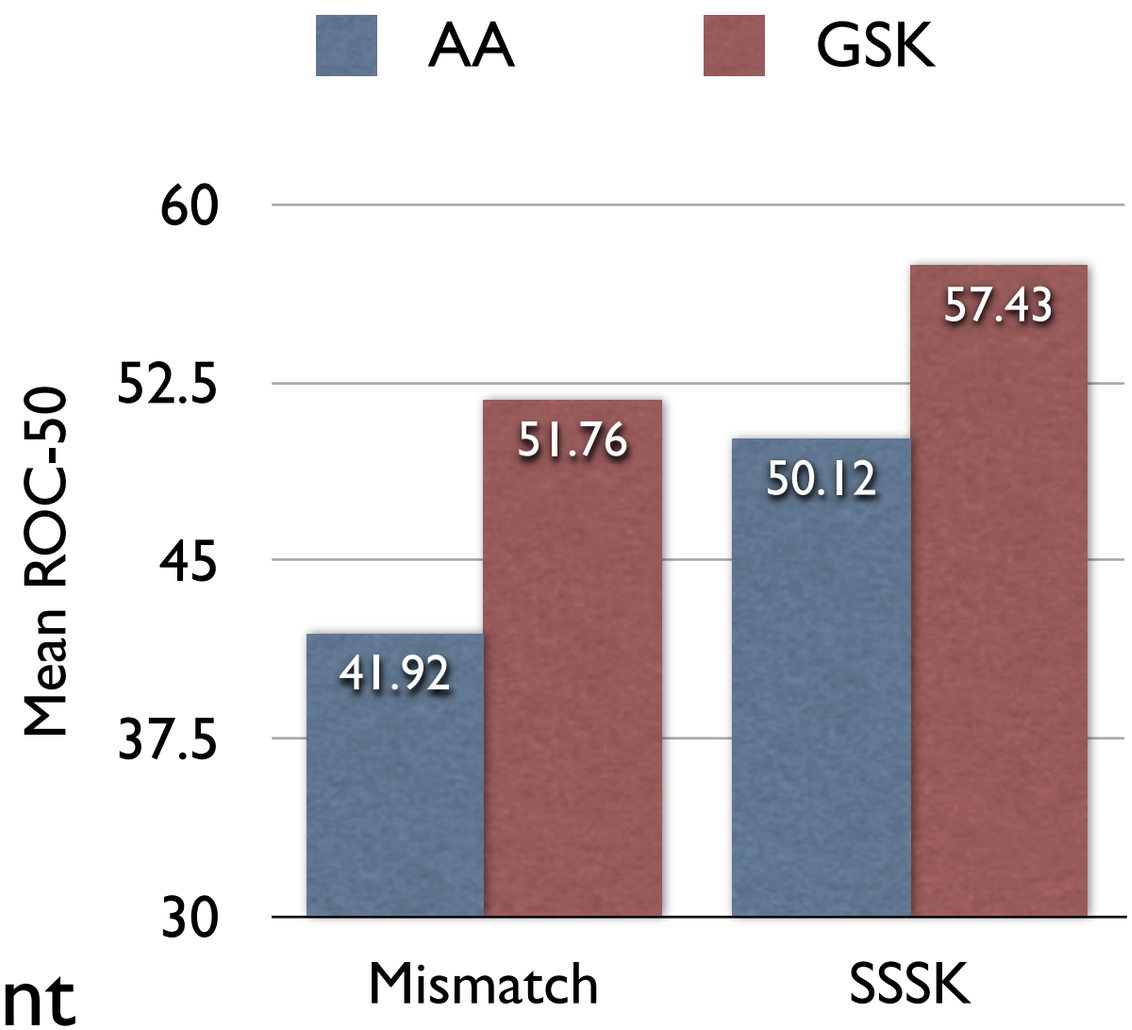
- Task 1: Protein remote homology prediction

Protein remote homology prediction

Data	Supervised setting	Semi-supervised setting
SCOP	7359 seqs (54 superfam)	+1M unlabeled seqs (NRDB)

- Each amino acid mapped into 20-dim BLOSUM descriptor
- Each protein is encoded as 8-dim sequence, $|E|=8$ bits using similarity hashing

Protein remote homology prediction



Improvement (GSK vs AA):

23%

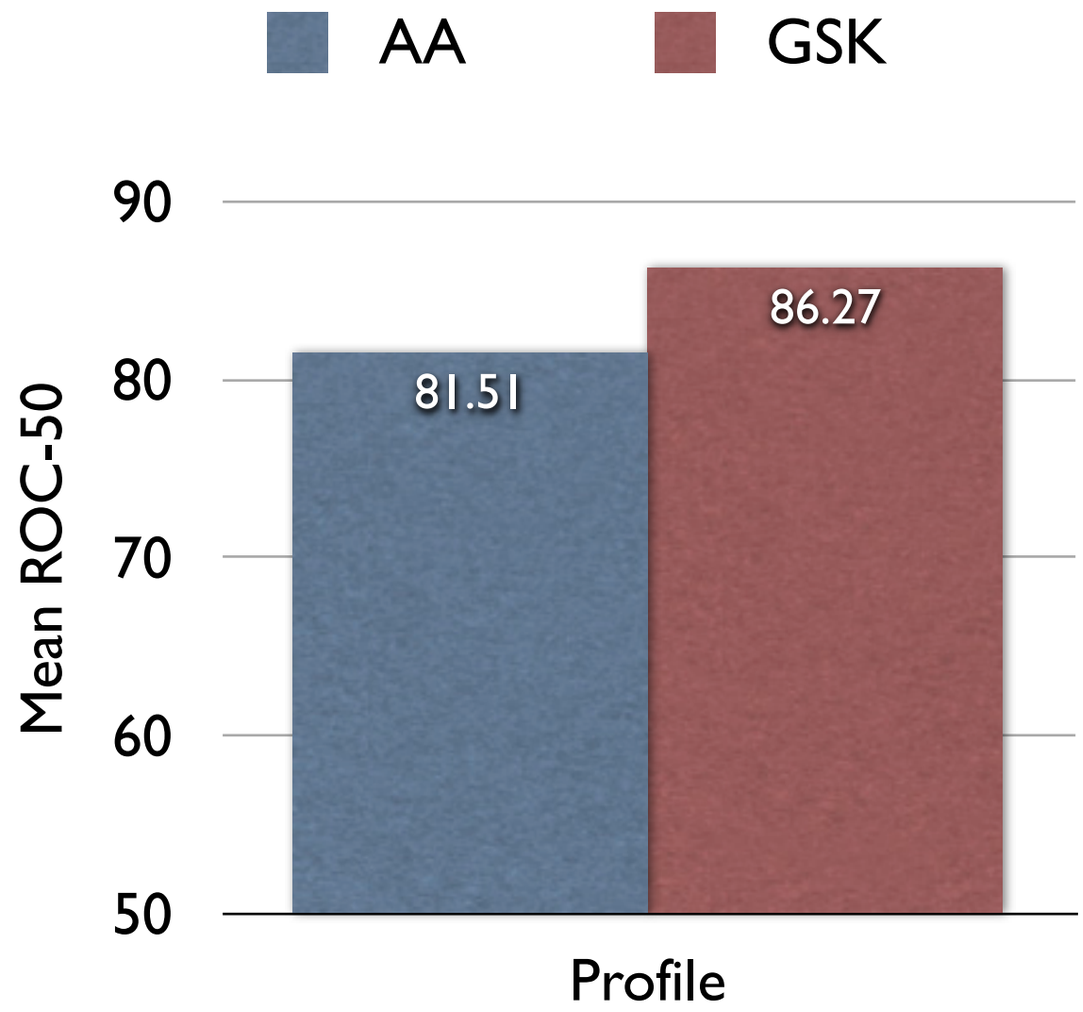
15%

AA = string kernel defined on amino acid sequences

GSK = general similarity kernel

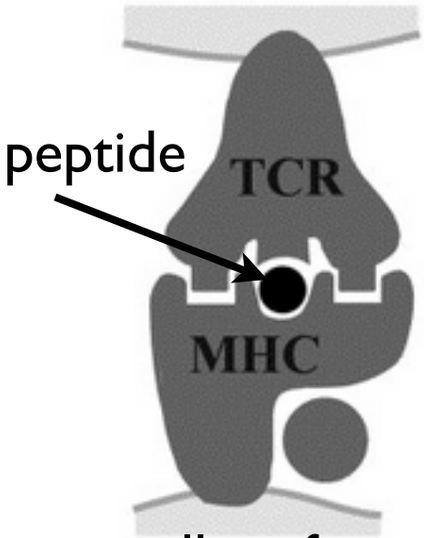
Protein remote homology prediction

Semi-supervised setting



- Task 2: MHC-peptide binding prediction

- MHC-Peptide binding prediction
  - needed for vaccine, immunotherapy design



Position	Peptide	binding strength
1	MVTEFIFLG	3.47
2	VTEFIFLGL	6.50
3	TEFIFLGLS	3.185
..	....	....
..	....	....

MVTEFIFLGLSD.....

antigen (sequence of target protein)

peptide prediction system output

MHC peptide binding prediction

- Data: tumor, cancer, virus antigens (WT1, PSA, EBV, etc)

Test ROC50 scores

Allele	QHMM	NetMHC	GSK
A0201	62.1	76.9	**82.34**
A0206	67.03	**83.93**	80.97
A2402	66.51	76.35	**85.24**

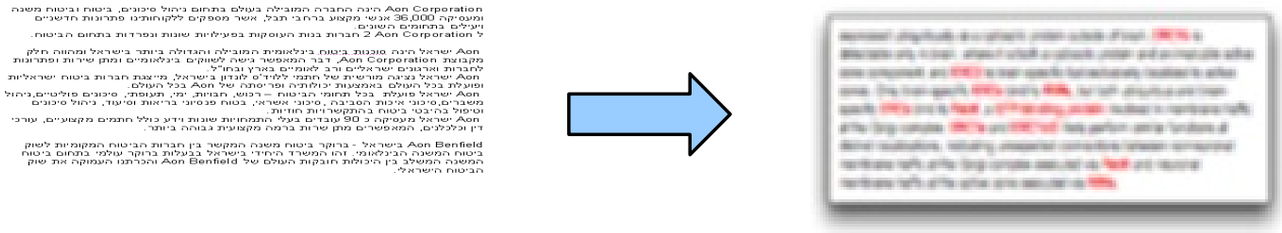
- QHMM = committee of HMMs
- NetMHC = highly optimized neural network predictor, trained on more data

Part III: Biomedical Literature Mining

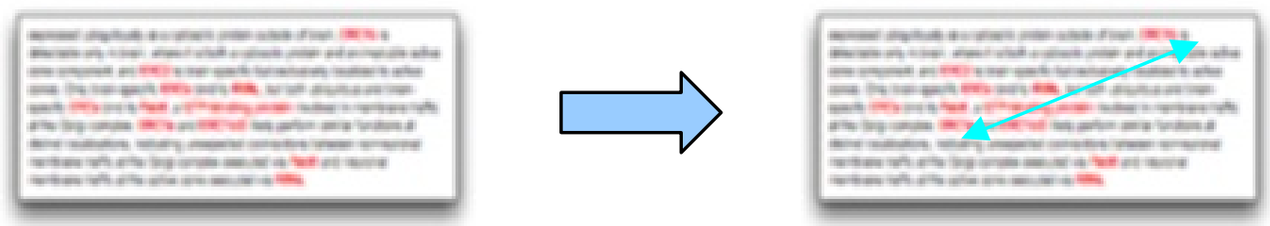
Practical information extraction/ retrieval problems in bioNLP

- Relevant article detection

- Bio-Entity Tagging (genes, proteins, etc)



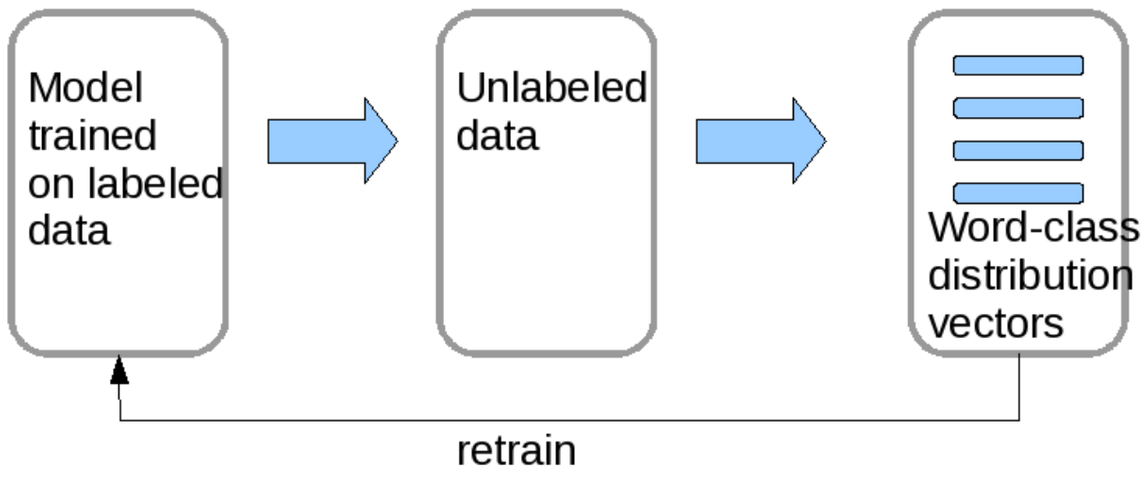
- Interaction/relationship (e.g., PPI) extraction



- Goal: automatic annotation, information extraction from biomedical literature

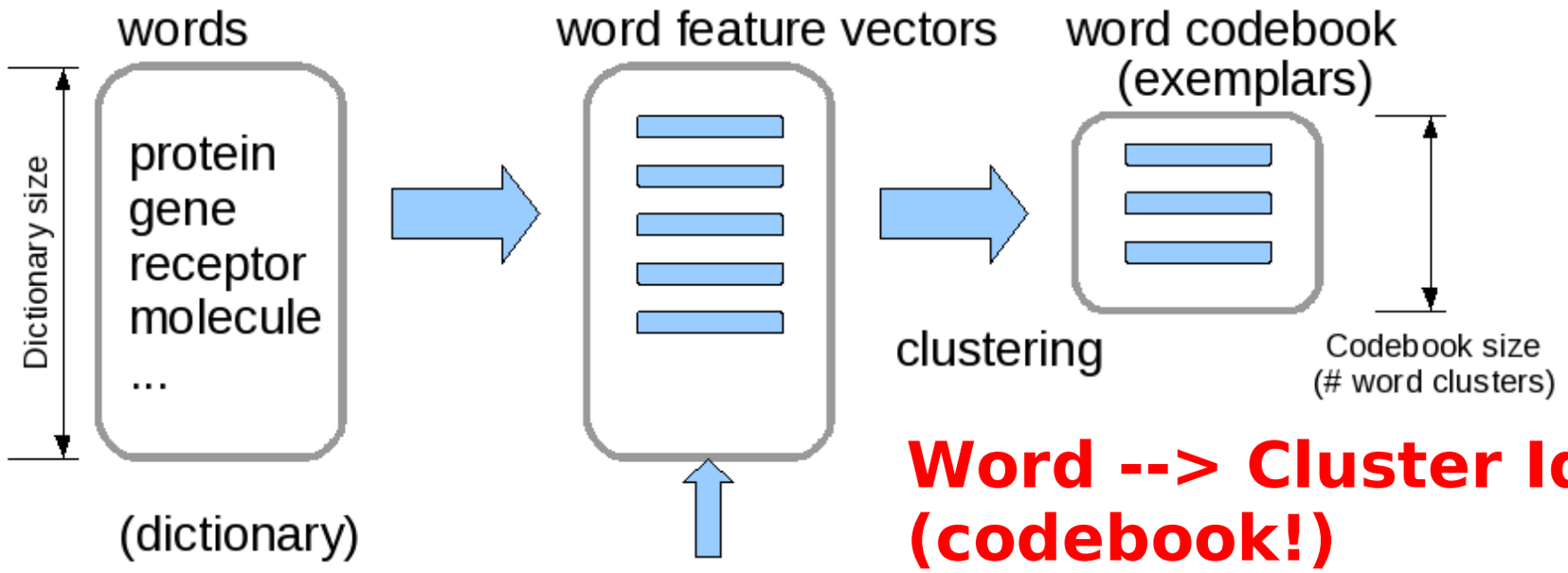
WCDL (Word-Class Distribution Learning)

- WCDL: simple and scalable semi-supervised feature learning method
  - Use model trained on labeled examples to estimate word-class distribution (WCD) on unlabeled data

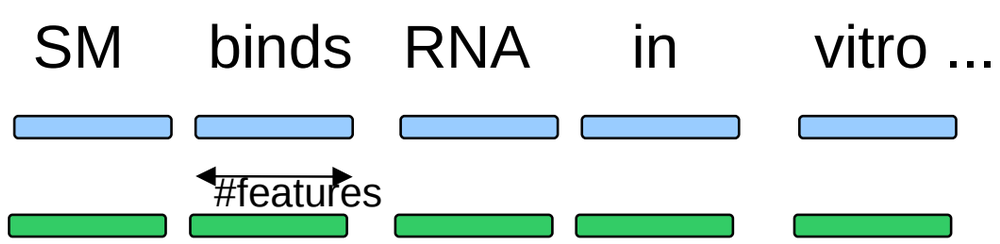


- Add WCD features to the feature set and retrain

Sequence Embeddings



word-class vectors (WCDL) language models



Gene Mention Prediction

- Find gene names in text:
  - Input text: Phenotypic analysis demonstrates that trio and Abl cooperate in regulating axon outgrowth...
  - Output gene names: trio, Abl
- Data: BioCreative II Competition
  - Train: 15K sentences
  - Test: 5K sentences
  - Unlabeled data for training: 60M sentences from PubMed (~1.3G words, approximately size of entire wikipedia)

Gene mention prediction

Method	Precision	Recall	F1
BioCreative II (best) semi-supervised method, extensive dictionaries	88.48	85.97	87.21
CRF (words)	87.84	76.92	82.02
CRF (WCDL)	90.70	85.19	**87.86**

PPI relation extraction

- Task: extract all interacting protein pairs from a given sentence
- Example:
  - **Input sentence:** The protein product of c-cbl proto-oncogene is known to ineract with several proteins, including Grb2, Crk, and PI3 kinase
  - Output: (c-cbl, Grb2), (c-cbl,Crk), etc

PPI relationship extraction

- Approach:

1. Learn word embeddings from unlabeled data (WCDL, LM)
2. Cluster words according to similarity S induced by learned embeddings
3. Map words to cluster ids w -> c(w)
4. Evaluate sentence similarity using cluster word ids (cluster kernel)

PPI relation extraction

Method	Precision	Recall	F1
Multiple kernel, multiple parser (Makoto et al) (state-of-the art method)	57.8	66.11	61.4
Dependency and deep parsers (Miyao et al)	54.9	65.5	59.5
Cluster kernel	60.68	69.08	**64.54**

PPI interaction article retrieval

- IAS task: classify article abstracts into protein-protein interaction (PPI)-related or not
  - Labeled data: 5495 abstracts for training / 677 for testing
  - Unlabeled data: 60M sentences from PubMed (~1.3G words)
    - use unlabeled data to construct word descriptors (ECML, 2010)

BioCreative II contest

- IAS test performance

method	F1	Accuracy
BioCreative II (best)	78.00	75.33
BioCreative II (rank 2)	77.95	77.10
String kernel (best)	77.17	74.30
WCDL (GSK)	**80.11**	**79.03**

- No complex hand-crafted features
- GSK + WCDL

Conclusions

- Sequence analysis space is highly non-trivial:
  - variable size (from short bio-molecules, proteins, to genomic DNA)
  - no all-purpose/generic representations
  - no standard way to do calculus
  - methods need to scale to very large datasets

- Need accurate yet efficiently computable representations and algorithms

Conclusions

- New, computationally efficient methods for k-mer sequence similarity models
- General Similarity models
- Semi-supervised learning method for annotation and relationship extraction
- Benefits: Accuracy + Efficiency

Future work

- Sequence typing (whole-genome analysis, read sets vs multilocus)
- Genes, proteins, small-molecules interaction
- Genome annotation
- Variant identification, finding functionally important variants from read data
- Dynamic change detection from massive read data (e.g., microbiome)

Future work

- Massive read sets analysis, comparison, annotation
- Whole-exome sequencing data analysis
- Semi-supervised kernel learning, learning from unlabeled data
- Efficient sequence labeling (annotation) methods

References

1. Pavel P. Kuksa, Pai-Hsi Huang, Vladimir Pavlovic. *Scalable Algorithms for String Kernels with Inexact Matching.* NIPS, 2008

2. Pavel P. Kuksa and Vladimir Pavlovic. *Efficient evaluation of large sequence kernels*. In *KDD*, 2012

3. Pavel P. Kuksa. *2D similarity kernels for biological sequence classification*. In *BIOKDD*, 2012.

4. Pavel P. Kuksa, Imdadullah Khan, and Vladimir Pavlovic. *Generalized Similarity Kernels for Efficient Sequence Classification*. In *SDM*, 2012.

5. Pavel P. Kuksa, Yanjun Qi, Bing Bai, Ronan Collobert, Jason Weston, Vladimir Pavlovic, and Xia Ning. *Semi-Supervised Abstraction-Augmented String Kernel for Multi-Level Bio-Relation Extraction*. In *ECML*, 2010

6. Pavel Kuksa and Vladimir Pavlovic. Efficient alignment-free DNA barcode analytics. *BMC Bioinformatics*, 10(Suppl 14):S9, 2009.

7. Pavel P. Kuksa. Biological Sequence Analysis with Multivariate String Kernels. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, March 2013.

8. Pavel Kuksa and Vladimir Pavlovic. Efficient motif finding algorithms for large-alphabet inputs. *BMC Bioinformatics*, 11(Suppl 8):S1, 2010.

9. Pavel P. Kuksa. Efficient sequence kernel-based genome-wide prediction of transcription factors. In *ICPR*, 2012.

10. Pavel Kuksa and Yanjun Qi. Semi-Supervised Bio-Named Entity Recognition with Word-Codebook Learning. In *SDM*, 2010.

11. Christina Leslie and Rui Kuang. *Fast string kernels using inexact matching for protein sequences*. JMLR, 2004

12. Pavel P. Kuksa and Vladimir Pavlovic. *Spatial Representation for Efficient Sequence Classification*. ICPR, 2010

13. Christina S. Leslie, Eleazar Eskin, Jason Weston, and William Stafford Noble. *Mismatch string kernels for svm protein classification*. NIPS , 2002

Thank you!