Apprentissage Automatique et Fouille de données textuelles

Jean-Michel RENDERS

Xerox Research Center Europe (France)

AAFD’06

Plan Global

- Introduction :
  - Fouille de textes
  - Spécificité des données textuelles
- Approche numéro 1 : méthodes à noyaux
  - Philosophie des méthodes à noyaux
  - Noyaux pour les données textuelles
- Approche numéro 2 : modèles génératifs
  - Génératif versus discriminatif – semi-supervisé
  - Modèles graphiques à variables latentes
  - Exemples : NB, PLSA, LDA, HPLSA
- Perspectives « récentes »

Fouille de Textes?

- Sens strict : très rare
- Sens large: contient une panoplie de sous-tâches
  - Recherche d'information (IR->QA)
  - Analyse sémantique
  - Catégorisation, Clustering
  - Extraction d'information → population d'ontologie
  - Focalisation utilisateur: navigation, visualisation, résumé adapté, traduction, …

  Souvent précédée de tâches de pré-traitement linguistique (jusqu'à l'analyse syntaxique et le tagging) … elles-mêmes appelées Fouille de textes!

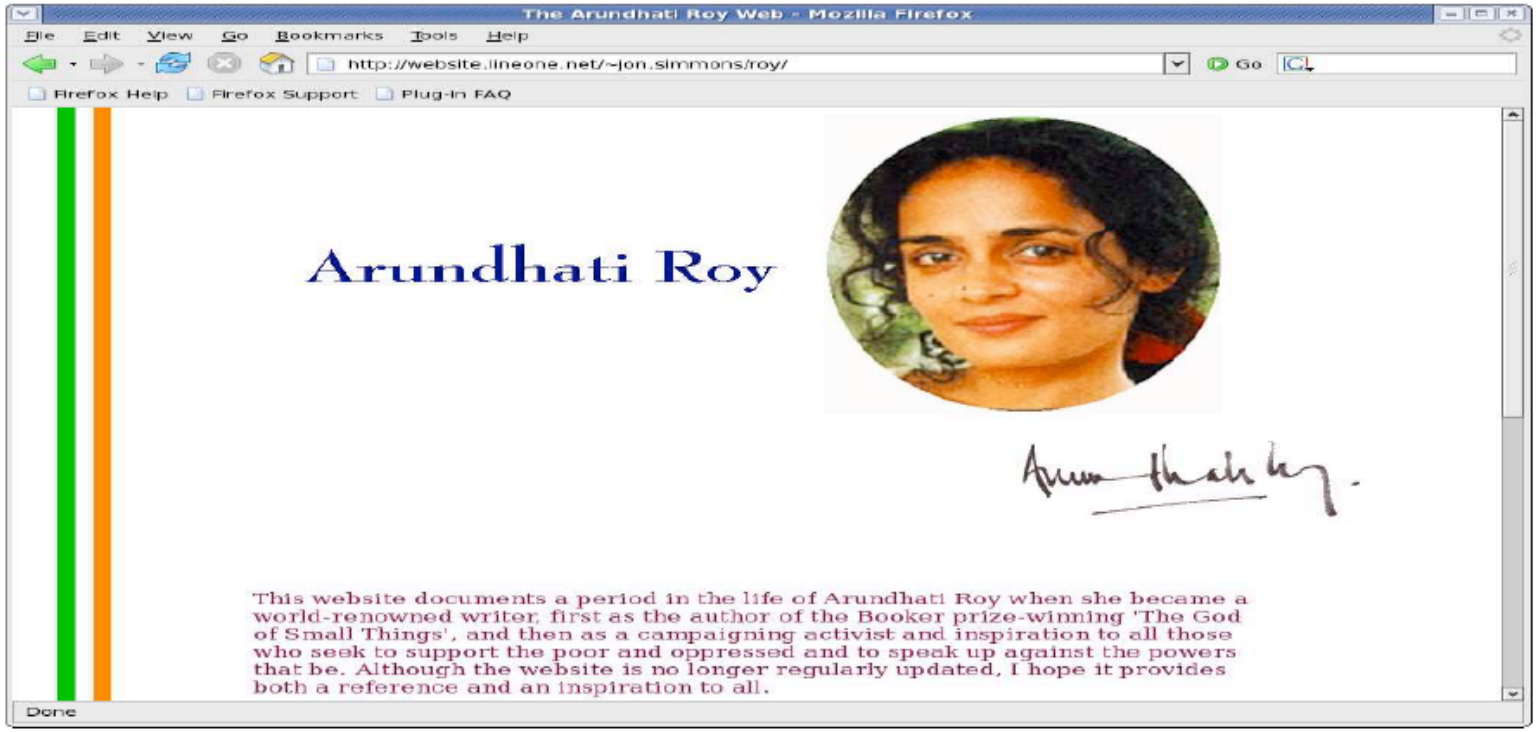
Spécificités du Texte

- Qu'est-ce qu'une observation?
  - Objet d'étude à différents niveaux de granularité (mot, phrase,section, document, corpus, mais aussi utilisateur, communauté)
- Lien entre forme et fond
  - Paradoxe structuré – non structuré
  - Importance d'un background knowledge
  - Redondance (cfr. Synonymie) et ambiguité (cfr. Polysémie)

Cas particulier

- Cas d'école le plus fréquent
  - Objet d'étude: document
  - Attributs: mots
- Propriétés:
  - Attributs: polysèmie, synonymie, structuration hiérarchique, dépendance ordonnée, attributs composés
  - Documents: polythématicité, structuration des classes, appartenance floue

Polythématicité



"This website documents a period in the life of Arundhati Roy when she became a world-renowned writer, first as the author of the Booker prize-winning *The God of Small Things*, and then as a campaigning activist and inspiration to all those who seek to support the poor and oppressed and to speak up against the power that be."

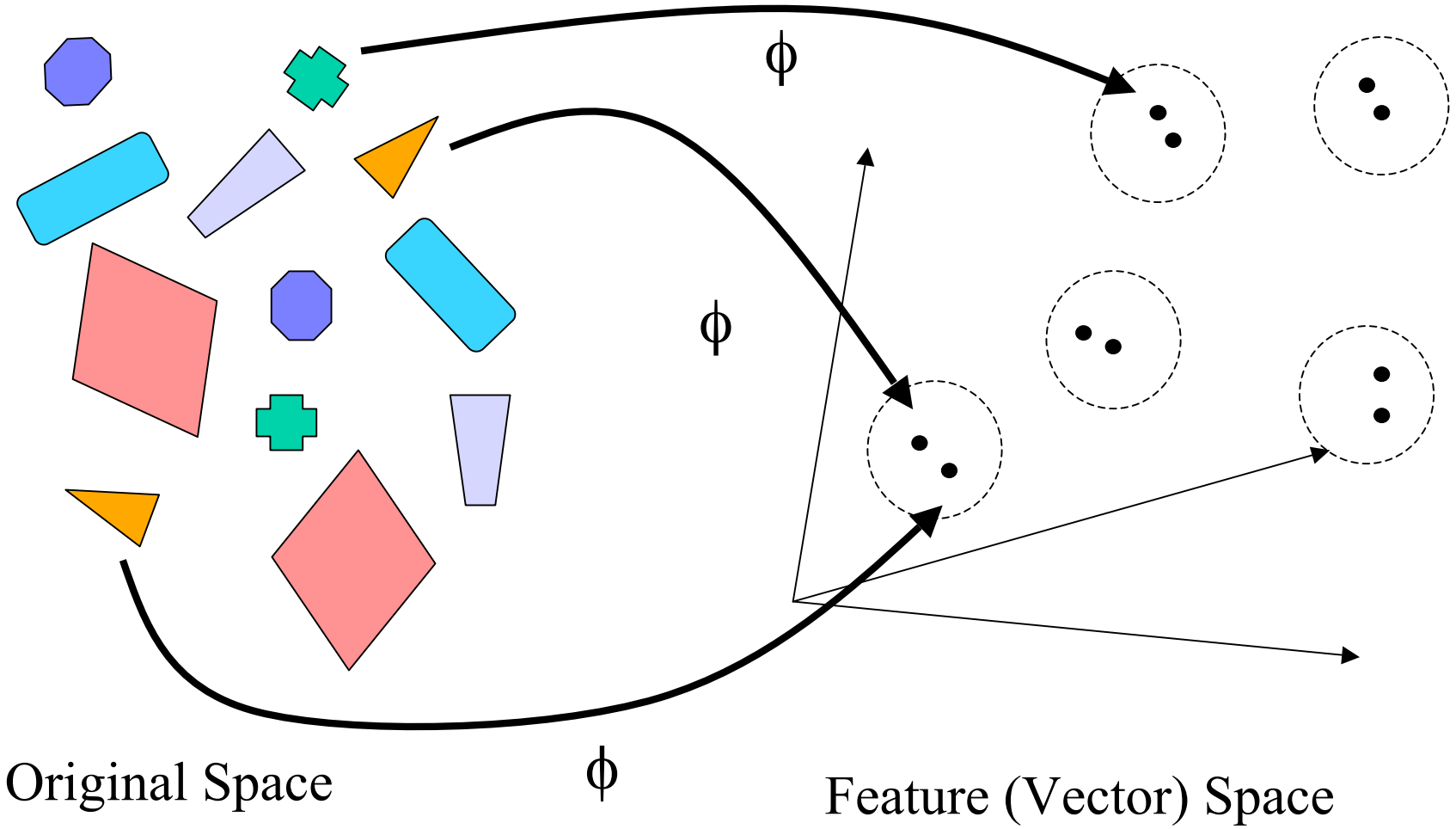
Approach 1 – Kernel Methods

- What's the philosophy of *Kernel Methods*?
- How to use Kernels Methods in *Learning* tasks?
- *Kernels for text* (BOW, latent concept, string, word sequence, tree and Fisher Kernels)
- *Applications* to NLP tasks

Kernel Methods : intuitive idea

- Find a mapping ϕ such that, in the new space, problem solving is easier (e.g. linear)
- The *kernel* represents the similarity between two objects (documents, terms, …), defined as the dot-product in this new vector space
- But the mapping is left implicit
- Easy generalization of a lot of dot-product (or distance) based pattern recognition algorithms

Kernel Methods : the mapping



Kernel : more formal definition

- A kernel k(x,y)
  - is a similarity measure
  - defined by an implicit mapping ϕ,
  - from the original space to a vector space (feature space)
  - such that: $k(x,y)=\phi(x)\bullet\phi(y)$
- This similarity measure and the mapping include:
  - Invariance or other a priori knowledge
  - Simpler structure (linear representation of the data)
  - The class of functions the solution is taken from
  - Possibly infinite dimension (hypothesis space for learning)
  - … but still computational efficiency when computing $k(x,y)$

Benefits from kernels

- Generalizes (nonlinearly) pattern recognition algorithms in clustering, classification, density estimation, …
  - When these algorithms are dot-product based, by replacing the dot product (x•y) by $k(x,y)=\phi(x)\bullet\phi(y)$

    e.g.: linear discriminant analysis, logistic regression, perceptron, SOM, PCA, ICA, …

  NM. This often implies to work with the "dual" form of the algo.
  - When these algorithms are distance-based, by replacing $d(x,y)$ by $k(x,x)+k(y,y)-2k(x,y)$
- Freedom of choosing ϕ implies a large variety of learning algorithms

Valid Kernels

- The function $k(x,y)$ is a valid kernel, if there exists a mapping ϕ into a vector space (with a dot-product) such that k can be expressed as $k(x,y)=\phi(x)\bullet\phi(y)$
- Theorem: $k(x,y)$ is a valid kernel if k is positive definite and symmetric (Mercer Kernel)
  - A function is P.D. if $\int K(\mathbf{x},\mathbf{y})f(\mathbf{x})f(\mathbf{y})d\mathbf{x}d\mathbf{y} \geq 0 \quad \forall f \in L_2$
  - In other words, the Gram matrix **K** (whose elements are $k(x_i,x_j)$) must be positive definite for all x_i, x_j of the input space
  - One possible choice of $\phi(x)$: $k(\bullet,x)$ (maps a point x to a function $k(\bullet,x)$ → feature space with infinite dimension!)

Example of Kernels (I)

- Polynomial Kernels: $k(x,y)=(x \bullet y)^d$
  - Assume we know most information is contained in monomials (e.g. multiword terms) of degree *d* (e.g. d=2: x_1^2, x_2^2, x_1x_2)
  - Theorem: the (implicit) feature space contains all possible monomials of degree d (ex: *n*=250; d=5; dim F=10^{10})
  - But kernel computation is only marginally more complex than standard dot product!
  - For $k(x,y)=(x \bullet y+1)^d$, the (implicit) feature space contains all possible monomials up to degree d !

The Kernel Gram Matrix

- With KM-based learning, the **sole** information used from the training data set is the Kernel Gram Matrix

$$K_{training} = \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & k(\mathbf{x}_1,\mathbf{x}_2) & \dots & k(\mathbf{x}_1,\mathbf{x}_m) \\ k(\mathbf{x}_2,\mathbf{x}_1) & k(\mathbf{x}_2,\mathbf{x}_2) & \dots & k(\mathbf{x}_2,\mathbf{x}_m) \\ \dots & \dots & \dots & \dots \\ k(\mathbf{x}_m,\mathbf{x}_1) & k(\mathbf{x}_m,\mathbf{x}_2) & \dots & k(\mathbf{x}_m,\mathbf{x}_m) \end{bmatrix}$$

- If the kernel is valid, K is symmetric definite-positive .

How to build new kernels

- Kernel combinations, preserving *validity*:

$$K(\mathbf{x},\mathbf{y}) = \lambda K_1(\mathbf{x},\mathbf{y}) + (1-\lambda)K_2(\mathbf{x},\mathbf{y}) \quad 0 \le \lambda \le 1$$

$$K(\mathbf{x},\mathbf{y}) = a.K_1(\mathbf{x},\mathbf{y}) \quad a > 0$$

$$K(\mathbf{x},\mathbf{y}) = K_1(\mathbf{x},\mathbf{y}).K_2(\mathbf{x},\mathbf{y})$$

$$K(\mathbf{x},\mathbf{y}) = f(x).f(y) \quad f \text{ is real-valued function}$$

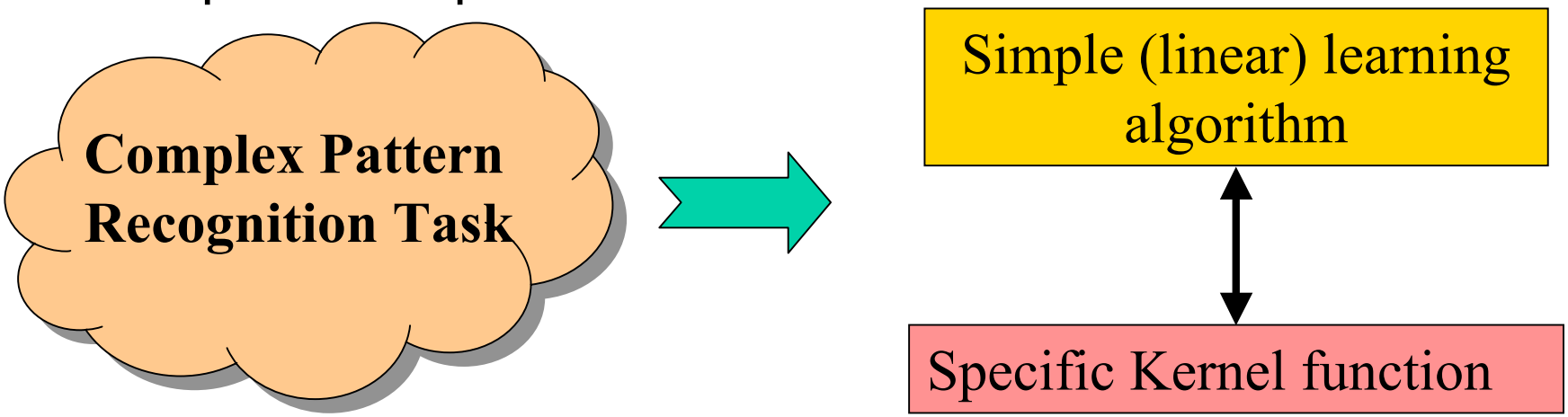
$$K(\mathbf{x},\mathbf{y}) = K_3(\ddot{\mathbf{o}}(\mathbf{x}),\ddot{\mathbf{o}}(\mathbf{y}))$$

$$K(\mathbf{x},\mathbf{y}) = \mathbf{x}'P\mathbf{y} \quad P \text{ symmetric definite positive}$$

$$K(\mathbf{x},\mathbf{y}) = \frac{K_1(\mathbf{x},\mathbf{y})}{\sqrt{K_1(\mathbf{x},\mathbf{x})}\sqrt{K_1(\mathbf{y},\mathbf{y})}}$$

Kernels and Learning

- In Kernel-based learning algorithms, problem solving is now decoupled into:
  - A general purpose learning algorithm (e.g. SVM, PCA, …) – Often linear algorithm (well-funded, robustness, …)
  - A problem specific kernel

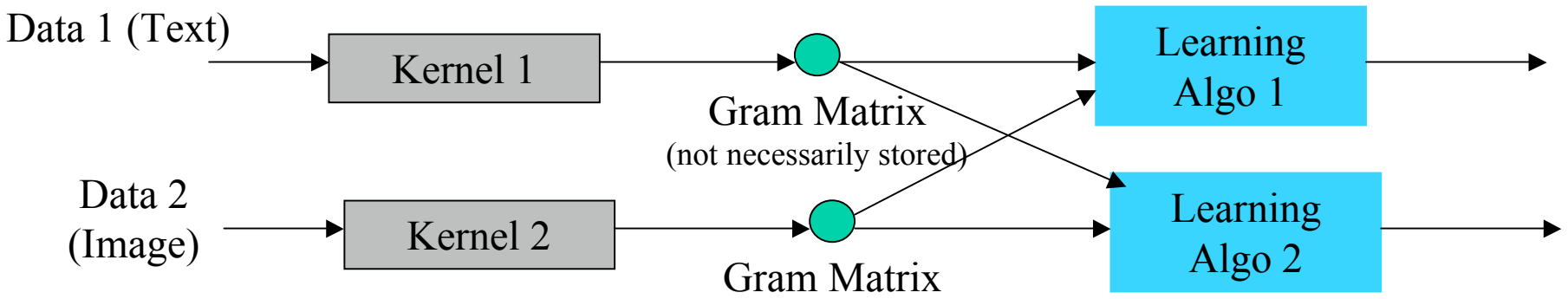


Learning in the feature space: Issues

- High dimensionality allows to render flat complex patterns by “explosion”
  - Computational issue, solved by designing kernels (efficiency in space and time)
  - Statistical issue (generalization), solved by the learning algorithm and also by the kernel
    - e.g. SVM, solving this complexity problem by maximizing the margin and the dual formulation
- E.g. RBF-kernel, playing with the σ parameter
- With adequate learning algorithms and kernels, high dimensionality is no longer an issue

Current Synthesis

- Modularity and re-usability
  - Same kernel ,different learning algorithms
  - Different kernels, same learning algorithms
- This presentation is allowed to focus only on designing kernels for textual data



Agenda

- What's the philosophy of *Kernel Methods*?
- How to use Kernels Methods in *Learning* tasks?
- *Kernels for text* (BOW, latent concept, string, word sequence, tree and Fisher Kernels)
- *Applications* to NLP tasks

Kernels for texts

- Similarity between documents?
  - Seen as ‘bag of words’ : dot product or polynomial kernels (multi-words)
  - Seen as set of concepts : GVSM kernels, Kernel LSI (or Kernel PCA), Kernel ICA, …possibly multilingual
  - Seen as string of characters: string kernels
  - Seen as string of terms/concepts: word sequence kernels
  - Seen as trees (dependency or parsing trees): tree kernels
  - Seen as the realization of probability distribution (generative model)

Strategies of Design

- Kernel as a way to encode prior information
  - Invariance: synonymy, document length, …
  - Linguistic processing: word normalisation, semantics, stopwords, weighting scheme, …
- Convolution Kernels: text is a recursively-defined data structure. How to build “global” kernels form local (atomic level) kernels?
- Generative model-based kernels: the “topology” of the problem will be translated into a kernel function (cfr. Mahalanobis)

Strategies of Design

- Kernel as a way to encode prior information
  - Invariance: synonymy, document length, …
  - Linguistic processing: word normalisation, semantics, stopwords, weighting scheme, …
- Convolution Kernels: text is a recursively-defined data structure. How to build “global” kernels form local (atomic level) kernels?
- Generative model-based kernels: the “topology” of the problem will be translated into a kernel function

‘Bag of words’ kernels (I)

- Document seen as a vector **d**, indexed by all the elements of a (controlled) dictionary. The entry is equal to the number of occurrences.
- A training corpus is therefore represented by a Term-Document matrix,

  noted $\mathbf{D}=[\mathbf{d}_1\ \mathbf{d}_2\ \ldots\ \mathbf{d}_{m-1}\ \mathbf{d}_m]$
- The “nature” of word: will be discussed later
- From this basic representation, we will apply a sequence of successive embeddings, resulting in a global (valid) kernel with all desired properties

BOW kernels (II)

- Properties:
  - All order information is lost (syntactical relationships, local context, …)
  - Feature space has dimension N (size of the dictionary)
- Similarity is basically defined by:

  $k(d_1,d_2)=\mathbf{d}_1 \cdot \mathbf{d}_2 = \mathbf{d}_1^t.\mathbf{d}_2$

  or, normalized (cosine similarity):

$$\hat{k}(d_1,d_2) = \frac{k(d_1,d_2)}{\sqrt{k(d_1,d_1).k(d_2,d_2)}}$$

- Efficiency provided by sparsity (and sparse dot-product algo): $O(|d_1|+|d_2|)$

‘Bag of words’ kernels: enhancements

- The choice of indexing terms:
  - Exploit linguistic enhancements:
    - Lemma / Morpheme & stem
    - Disambiguised lemma (lemma+POS)
    - Noun Phrase (or useful collocation, n-grams)
    - Named entity (with type)
  - Exploit IR lessons
    - Stopword removal
    - Feature selection based on frequency
    - Weighting schemes (e.g. idf)
    - Semantic enrichment by term-term similarity matrix **Q** (positive definite): $k(d_1,d_2)=\phi(\mathbf{d}_1)^t.\mathbf{Q}.\phi(\mathbf{d}_2)$
  - NB. Using polynomial kernels up to degree *p*, is a natural and efficient way of considering all (up-to-)*p*-grams (with different weights actually), but order is not taken into account (“sinking ships” is the same as “shipping sinks”)

Semantic Smoothing Kernels

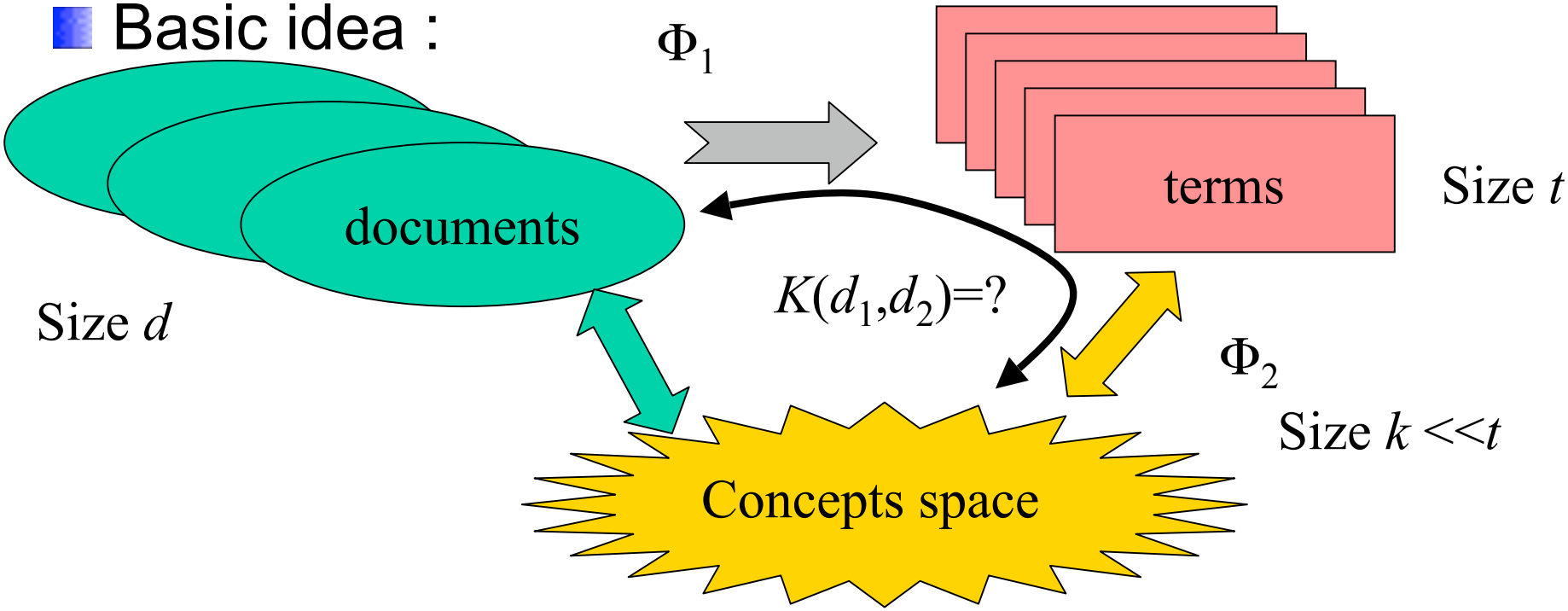
- Synonymy and other term relationships:
  - GVSM Kernel: the term-term co-occurrence matrix ($\mathbf{DD}^t$) is used in the kernel: $k(d_1,d_2)=\mathbf{d}_1^t.(\mathbf{D}.\mathbf{D}^t).\mathbf{d}_2$
  - The completely kernelized version of GVSM is:
    - The training kernel matrix $\mathbf{K}(=\mathbf{D}^t.\mathbf{D}) \rightarrow \mathbf{K}^2$ (mxm)
    - The kernel vector of a new document *d* vs the training documents : $\mathbf{t} \rightarrow \mathbf{K.t}$ (mx1)
    - The initial **K** could be a polynomial kernel (GVSM on multi-words terms)
  - Variants: One can use
    - a shorter context than the document to compute term-term similarity (term-context matrix)
    - Another measure than the number of co-occurrences to compute the similarity (e.g. Mutual information, …)
  - Can be generalised to $\mathbf{K}^n$ (or a weighted combination of $\mathbf{K}^1\ \mathbf{K}^2 \ldots \mathbf{K}^n$ cfr. Diffusion kernels later), but is $\mathbf{K}^n$ less and less sparse! Interpretation as sum over paths of length 2n.

Semantic Smoothing Kernels

- Can use other term-term similarity matrix than $\mathbf{DD}^t$; e.g. a similarity matrix derived from the Wordnet thesaurus, where the similarity between two terms is defined as:
  - the inverse of the length of the path connecting the two terms in the hierarchical hyper/hyponymy tree.
  - A similarity measure for nodes on a tree (feature space indexed by each node *n* of the tree, with ϕ_n(term x) if term *x* is the class represented by *n* or "under" *n*), so that the similarity is the number of common ancestors (including the node of the class itself).
- With semantic smoothing, 2 documents can be similar even if they don't share common words.

Latent concept Kernels

- Basic idea :

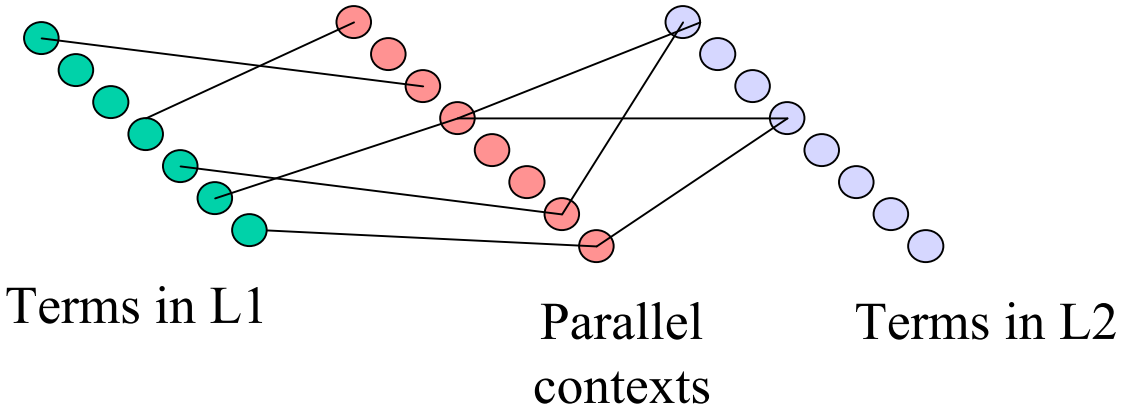


Latent concept Kernels

- $k(d_1,d_2)=\phi(\mathbf{d}_1)^t.\mathbf{P}^t.\mathbf{P}.\phi(\mathbf{d}_2)$,
  - where **P** is a (linear) projection operator
    - From Term Space
    - to Concept Space
- Working with (latent) concepts provides:
  - Robustness to polysemy, synonymy, style, …
  - Cross-lingual bridge
  - Natural Dimension Reduction
- But, how to choose **P** and how to define (extract) the latent concept space? Ex: Use PCA : the concepts are nothing else than the principal components.

Why multilingualism helps …

- Graphically:



- Concatenating both representations will force language-independent concept: each language imposes constraints on the other
- Searching for maximally correlated projections of paired observations (CCA) has a sense, semantically speaking

Diffusion Kernels

- Recursive dual definition of the semantic smoothing:

  $\mathbf{K}=\mathbf{D}'(\mathbf{I}+u\mathbf{Q})\mathbf{D}$

  $\mathbf{Q}=\mathbf{D}(\mathbf{I}+v\mathbf{K})\mathbf{D}'$

  NB. $u=v=0$ → standard BOW; $v=0$ → GVSM

- Let B= **D'D** (standard BOW kernel); **G=DD'**

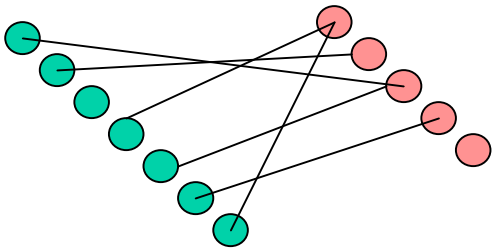
- If $u=v$, The solution is the "Von Neumann diffusion kernel"

  - $\mathbf{K}=\mathbf{B}.(\mathbf{I}+u\mathbf{B}+u^2\mathbf{B}^2+\ldots)=\mathbf{B}(\mathbf{I}-u\mathbf{B})^{-1}$ and $\mathbf{Q}=\mathbf{G}(\mathbf{I}-u\mathbf{G})^{-1}$ [only of $u<||\mathbf{B}||^{-1}$]
  - Can be extended, with a faster decay, to exponential diffusion kernel: $\mathbf{K}=\mathbf{B}.\exp(u\mathbf{B})$ and $\mathbf{Q}=\exp(u\mathbf{G})$

Graphical Interpretation

- These diffusion kernels correspond to defining similarities between nodes in a graph, specifying only the myopic view

Documents

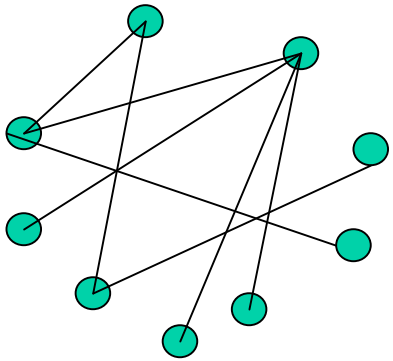


The (weighted) adjacency matrix is the Doc-Term Matrix

Terms

Or

By aggregation, the (weighted) adjacency matrix is the term-term similarity matrix **G**



Diffusion kernels corresponds to considering all paths of length 1, 2, 3, 4 … linking 2 nodes and summing the product of local similarities, with different decay strategies

It is in some way similar to KPCA by just "rescaling" the eigenvalues of the basic Kernel matrix (decreasing the lowest ones)

Strategies of Design

- Kernel as a way to encode prior information
  - Invariance: synonymy, document length, …
  - Linguistic processing: word normalisation, semantics, stopwords, weighting scheme, …
- Convolution Kernels: text is a recursively-defined data structure. How to build “global” kernels form local (atomic level) kernels?
- Generative model-based kernels: the “topology” of the problem will be translated into a kernel function

Sequence kernels

- Consider a document as:
  - A sequence of characters (string)
  - A sequence of tokens (or stems or lemmas)
  - A paired sequence (POS+lemma)
  - A sequence of concepts
  - A tree (parsing tree)
  - A dependency graph

  (later)
- Sequence kernels → order has importance
  - Kernels on string/sequence : counting the subsequences two objects have in common … but various ways of counting
  - Contiguity is necessary (p-spectrum kernels)
  - Contiguity is not necessary (subsequence kernels)
  - Contiguity is penalised (gap-weighted subsequence kernels)

String and Sequence

- Just a matter of convention:
  - String matching: implies contiguity
  - Sequence matching : only implies order

Gap-weighted subsequence kernels

- Feature space indexed by all elements of Σ^p
- $\phi_u(s)$=sum of weights of occurrences of the p-gram u as a (non-contiguous) subsequence of s, the weight being length penalizing: $\lambda^{length(u)}$) [NB: length includes both matching symbols and gaps]
- Example:
  - D1 : ATCGTAGACTGTC
  - D2 : GACTATGC
  - $(D1)_{CAT} = 2\lambda^8+2\lambda^{10}$ and $(D2)_{CAT} = \lambda^4$
  - $k(D1,D2)_{CAT}=2\lambda^{12}+2\lambda^{14}$
- Naturally built as a dot product → valid kernel
- For alphabet of size 80, there are 512000 trigrams
- For alphabet of size 26, there are 12.10^6 5-grams

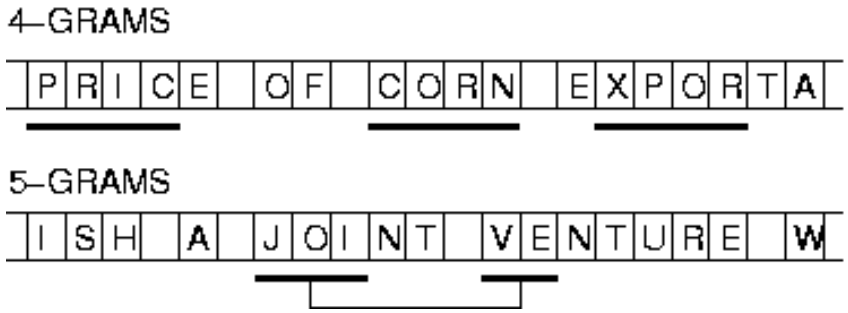
Gap-weighted subsequence kernels

- Hard to perform explicit expansion and dot-product!
- Efficient recursive formulation (dynamic programming –like), whose complexity is $O(k.|D1|.|D2|)$
- Normalization (doc length independence)

$$\hat{k}(d_1, d_2) = \frac{k(d_1, d_2)}{\sqrt{k(d_1, d_1).k(d_2, d_2)}}$$

Word Sequence Kernels (I)

- Here “words” are considered as symbols
  - Meaningful symbols → more relevant matching
  - Linguistic preprocessing can be applied to improve performance
  - Shorter sequence sizes → improved computation time
  - But increased sparsity (documents are more : “orthogonal”)
  - Intermediate step: syllable kernel (indirectly realizes some low-level stemming and morphological decomposition)
- Motivation : the noisy stemming hypothesis (important N-grams approximate stems), confirmed experimentally in a categorization task



Word Sequence Kernels (II)

- Link between Word Sequence Kernels and other methods:
  - For *k*=1, WSK is equivalent to basic “Bag Of Words” approach
  - For λ=1, close relation to polynomial kernel of degree k, WSK takes order into account
- Extension of WSK:
  - Symbol dependant decay factors (way to introduce IDF concept, dependence on the POS, stop words)
  - Different decay factors for gaps and matches (e.g. $\lambda_{noun}<\lambda_{adj}$ when gap; $\lambda_{noun}>\lambda_{adj}$ when match)
  - Soft matching of symbols (e.g. based on thesaurus, or on dictionary if we want cross-lingual kernels)

Trie-based kernels

- An alternative to DP based on string matching techniques
- TRIE= Retrieval Tree (cfr. Prefix tree) = tree whose internal nodes have their children indexed by Σ.
- Suppose F= Σ^p : the leaves of a complete p-trie are the indices of the feature space
- Basic algorithm:
  - Generate all substrings s(i:j) satisfying initial criteria; idem for t.
  - Distribute the s-associated list down from root to leave (depth-first)
  - Distribute the t-associated list down from root to leave taking into account the distribution of s-list (pruning)
  - Compute the product at the leaves and sum over the leaves
- Key points: in steps (2) and (3), not all the leaves will be populated (else complexity would be O(| Σ^p|) … you need not build the trie explicitly!

Tree Kernels

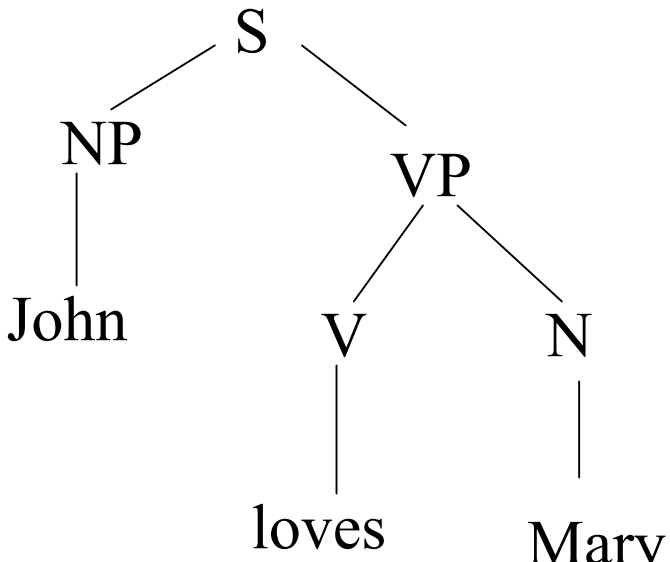
- Application: categorization [one doc=one tree], parsing (desambiguation) [one doc = multiple trees]
- Tree kernels constitute a particular case of more general kernels defined on discrete structure (convolution kernels). Intuitively, the philosophy is
  - to split the structured objects in parts,
  - to define a kernel on the “atoms” and a way to recursively combine kernel over parts to get the kernel over the whole.

Fundaments of Tree kernels

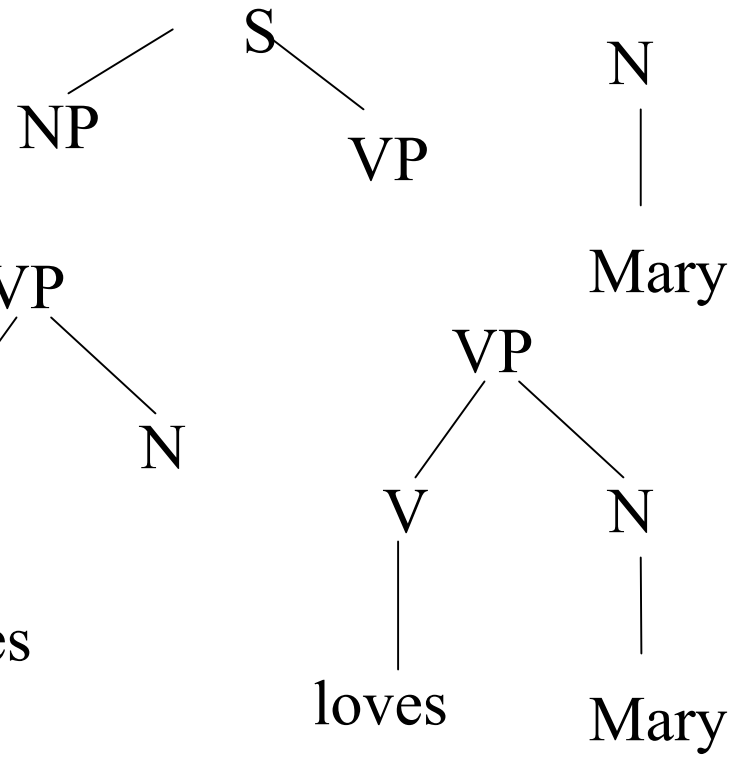
- Feature space definition: one feature for each possible proper subtree in the training data; feature value = number of occurences
- A subtree is defined as any part of the tree which includes more than one node, with the restriction there is no “partial” rule production allowed.

Tree Kernels : example

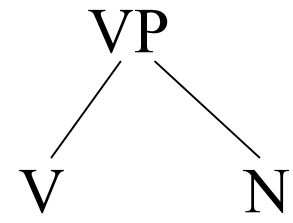
- Example :



A Parse Tree



… a few among the many subtrees of this tree!



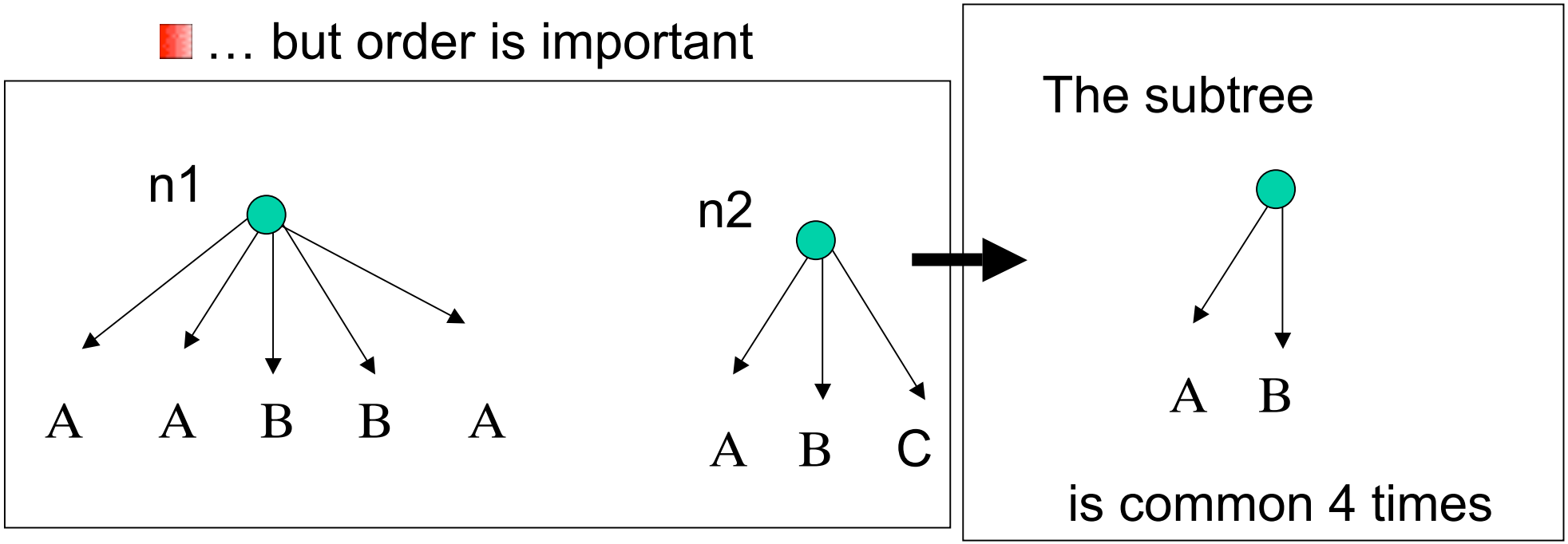
Tree Kernels : algorithm

- Kernel = dot product in this high dimensional feature space
- Once again, there is an efficient recursive algorithm (in polynomial time, not exponential!)
- Basically, it compares the production of all possible pairs of nodes (n1,n2) (n1∈T1, n2 ∈ T2); if the production is the same, the number of common subtrees routed at both n1 and n2 is computed recursively, considering the number of common subtrees routed at the common children
- Formally, let $k_{co\text{-}rooted}$(n1,n2)=number of common subtrees rooted at both n1 and n2

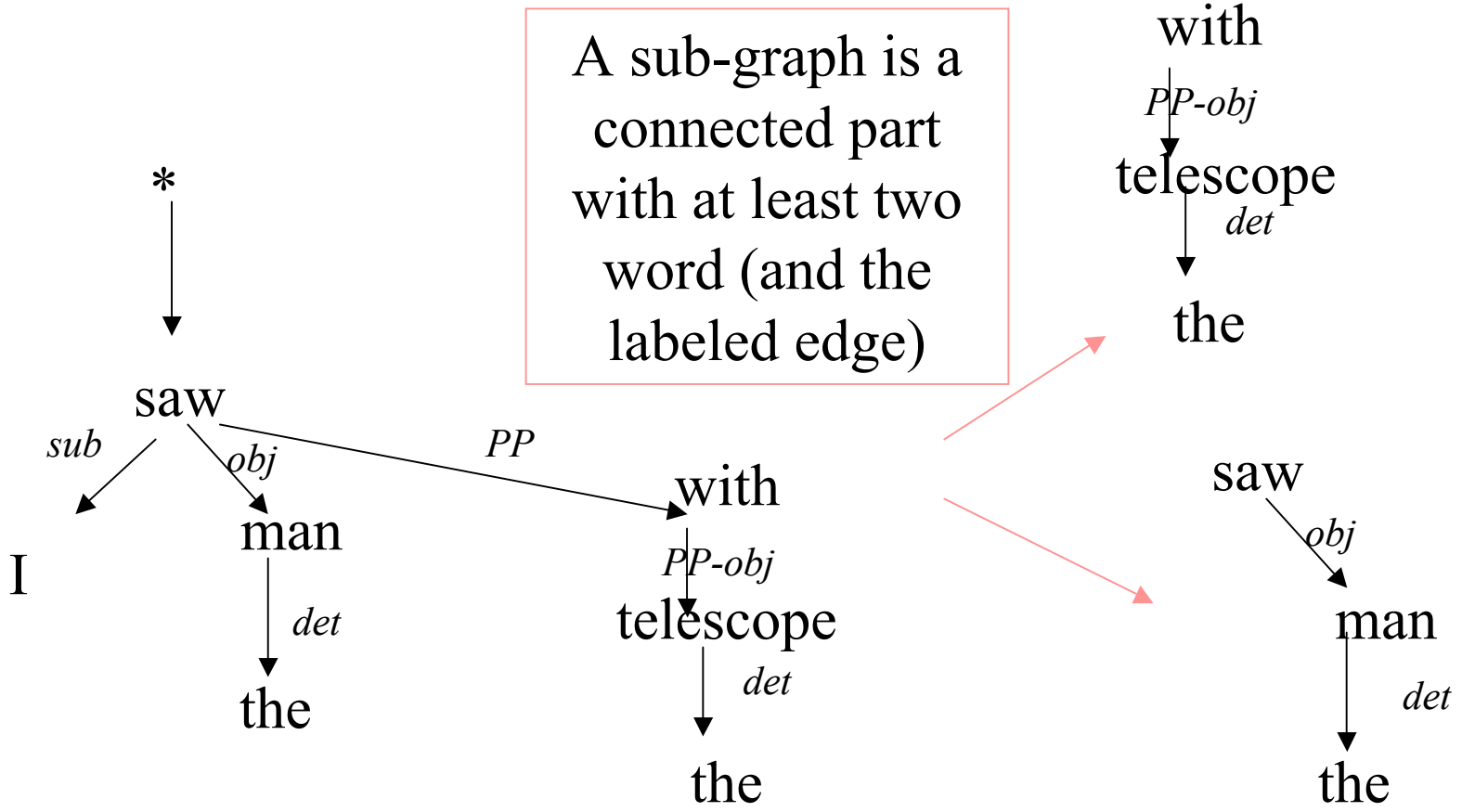
$$k(T_1, T_2) = \sum_{n_1 \in T_1} \sum_{n_2 \in T_2} k_{co-rooted}(n_1, n_2)$$

Variant for labeled ordered tree

- Example: dealing with html/xml documents
- Extension to deal with:
  - Partially equal production
  - Children with same labels
  - … but order is important

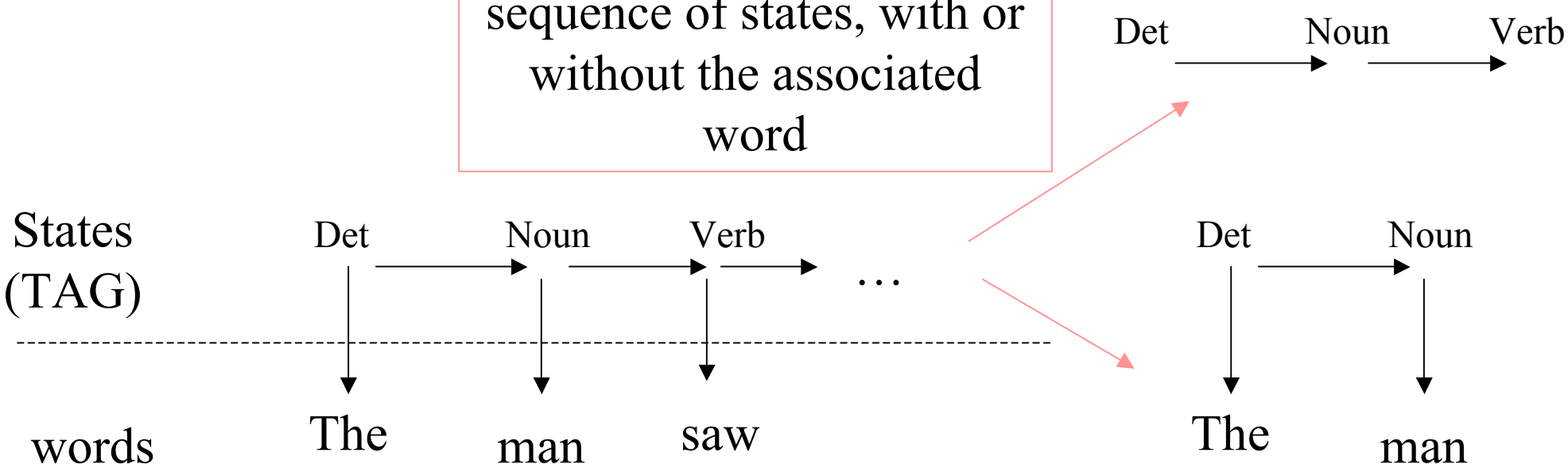


Dependency Graph Kernel



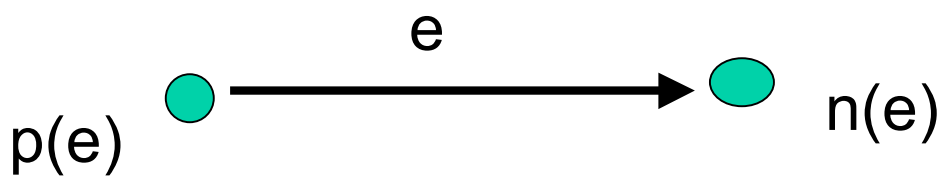
Paired sequence kernel

A subsequence is a sub-sequence of states, with or without the associated word



Graph kernels based on Common Walks

- Walk = (possibly infinite) sequence of labels obtained by following edges on the graph
- Path = walk with no vertex visited twice
- Important concept: direct product of two graphs G1xG2
  - V(G1xG2)={(v1,v2), v1 and v2: same labels)
  - E(G1xG2)={(e1,e2): e1, e2: same labels, p(e1) and p(e2) same labels, n(e1) and n(e2) same labels}



Strategies of Design

- Kernel as a way to encode prior information
  - Invariance: synonymy, document length, …
  - Linguistic processing: word normalisation, semantics, stopwords, weighting scheme, …
- Convolution Kernels: text is a recursively-defined data structure. How to build "global" kernels form local (atomic level) kernels?
- Generative model-based kernels: the "topology" of the problem will be translated into a kernel function

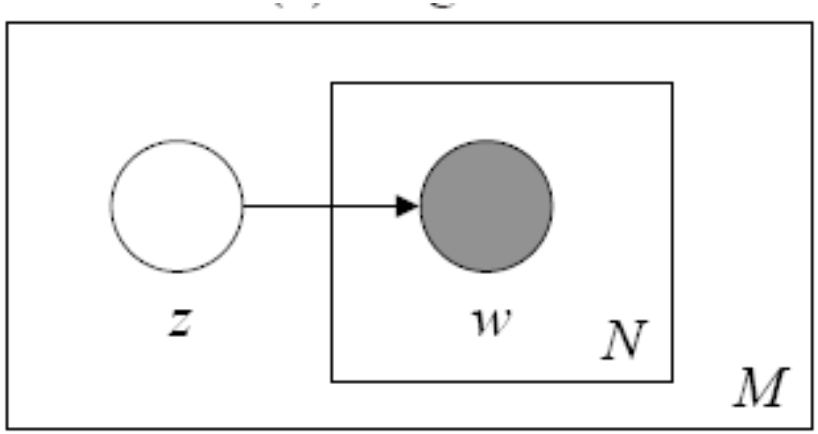
Plan Global

- Introduction :
  - Fouille de textes
  - Spécificité des données textuelles
- Approche numéro 1 : méthodes à noyaux
  - Philosophie des méthodes à noyaux
  - Noyaux pour les données textuelles
- Approche numéro 2 : modèles génératifs
  - Génératif versus discriminatif – semi-supervisé
  - Modèles graphiques à variables latentes
  - Exemples : NB, PLSA, LDA, HPLSA
- Perspectives « récentes »

Generative vs Discriminative

- Generative approach:
  - Model P(x,y) (= P(y|x).P(x) = P(x|y).P(y))
  - Then, for a new x, choose y = argmax P(x,y)
- Discriminative approach:
  - Model P(y|x)
  - Then, for a new x, choose y = argmax P(y|x)
- Most advantages for discriminative approach but:
  - Semi-supervised learning – continuum between clustering and categorization
  - Novelty Detection
  - NB. Most generative approaches use latent variables (hidden classes or components) – strong link between component and categories – Then use probabilistic values of these latent variables as new features in a discriminative setting (cfr. Dimension reduction – generative model-based kernels)

Graphical models : NB

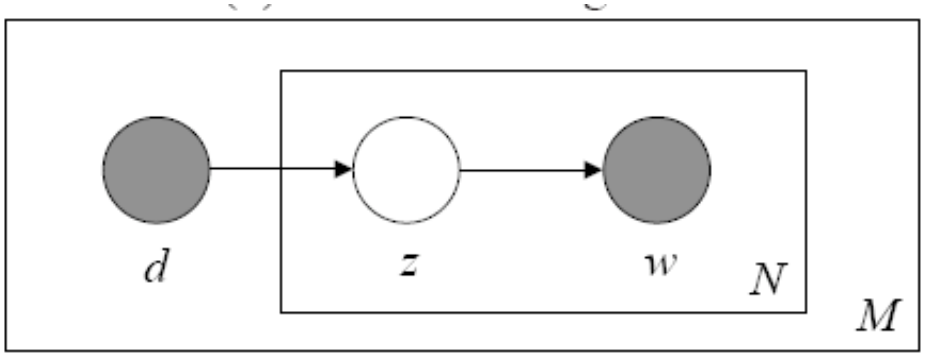


$$p(\mathbf{w}) = \sum_{z} p(z) \prod_{n=1}^{N} p(w_n \mid z).$$

- M documents
- N words
- 1! Topic per document
- Supervised case (z observed):
  - Training : Parameters (class priors and class profiles) by max likelihood
  - Classify : max p(**w**,z)
- Unsupervised:

  Use EM

PLSA

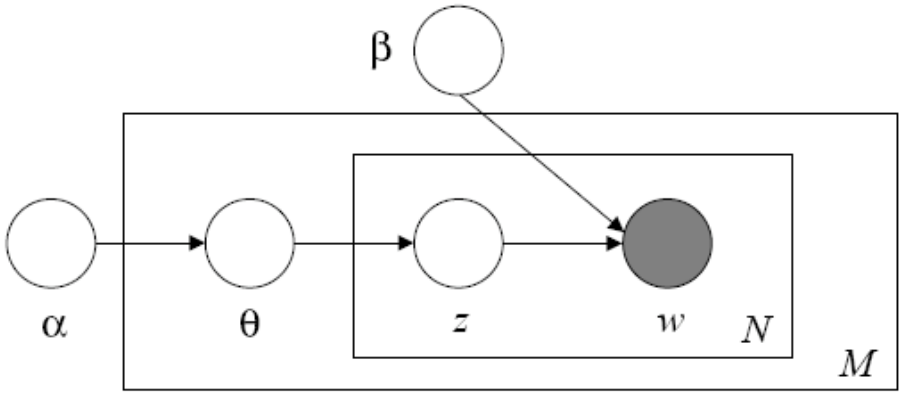


$$p(d, w_n) = p(d) \sum_z p(w_n | z) p(z | d).$$

- M documents
- N words
- Multiple Topics per document
- Supervised case
  - Parameters (p(z,d) and class profiles) by max likelihood
  - Inference : by EM to identify p(z|d)
- Unsupervised:

  Use tempered-EM

LDA



$$p(\mathbf{w}|\alpha,\beta) = \int p(\theta|\alpha)\left(\prod_{n=1}^{N}\sum_{z_n} p(z_n|\theta)p(w_n|z_n,\beta)\right)d\theta.$$

- M documents
- N words
- Multiple Topics per document
- Dirichlet prior on the topic mixing proportion
- Supervised case
  - Parameters (α,β) (class priors and class profiles) by max likelihood, given w, θ,z
  - Variational Inference : to identify p(θ,z| α,β,**w**)
- Unsupervised:

  Use variational-EM to identify (α,β) , given observed **w**

"Arts"	"Budgets"	"Children"	"Education"
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

Strategies of Design

- Kernel as a way to encode prior information
  - Invariance: synonymy, document length, …
  - Linguistic processing: word normalisation, semantics, stopwords, weighting scheme, …
- Convolution Kernels: text is a recursively-defined data structure. How to build "global" kernels form local (atomic level) kernels?
- Generative model-based kernels: the "topology" of the problem will be translated into a kernel function

Remind

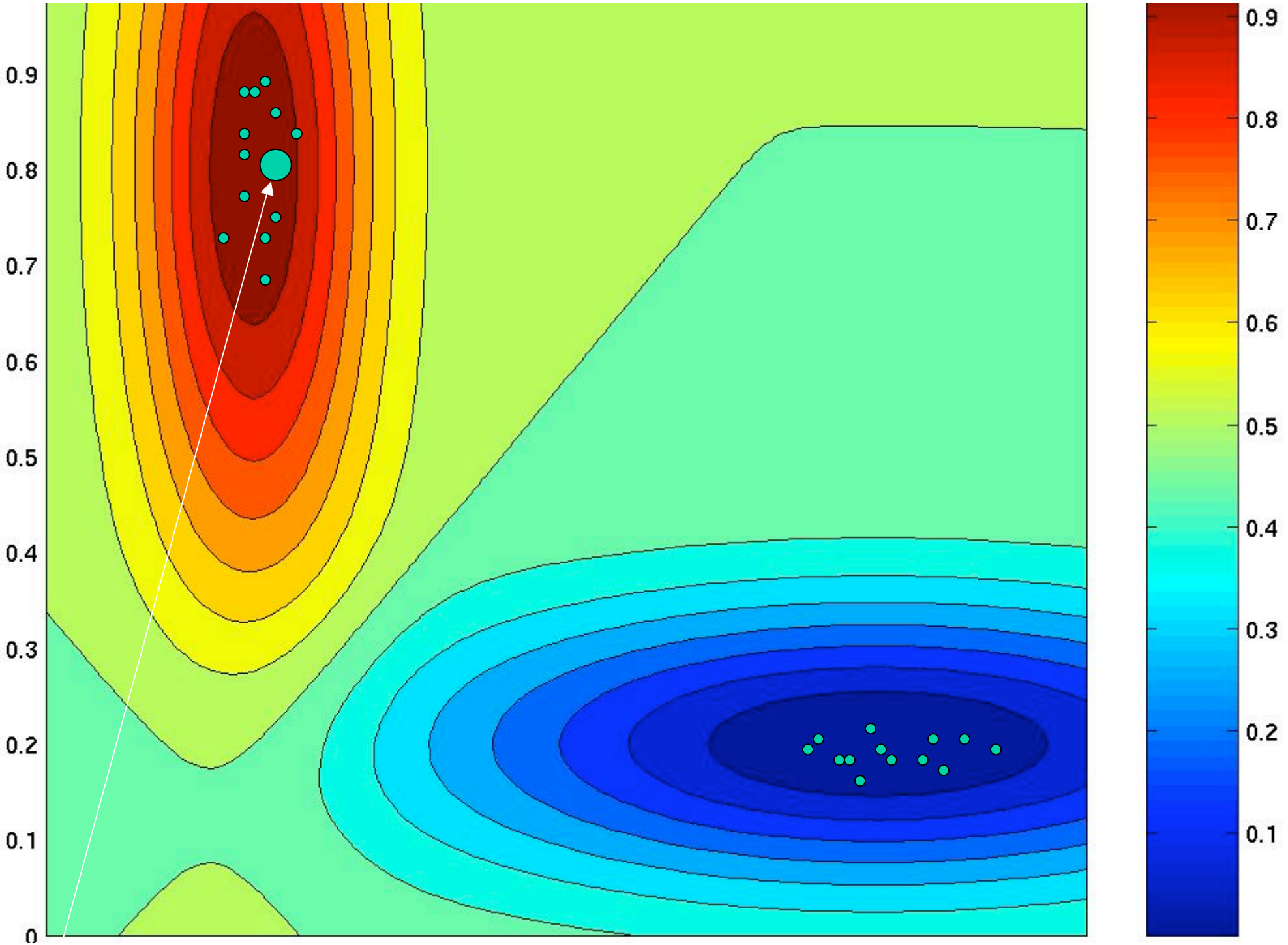
- This family of strategies brings you the additional advantage of using all your unlabeled training data to design more problem-adapted kernels
- They constitute a natural and elegant way of solving semi-supervised problems (mix of labelled and unlabelled data)

Marginalised – Conditional Independence Kernels

- Assume a family of models M (with prior p0(m) on each model) [finite or countably infinite]
- each model m gives P(x|m)
- Feature space indexed by models: x→ P(x|m)
- Then, assuming conditional independence, the joint probability is given by

$$P_M(x,z) = \sum_{m \in M} P(x,z \mid m) P_0(m) = \sum_{m \in M} P(x \mid m) P(z \mid m) P_0(m)$$

- This defines a valid probability-kernel (CI implies PD kernel), by marginalising over m. Indeed, the gram matrix is K=P.diag(P0).P’ (some reminiscence of latent concept kernels)


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1

Fisher Kernels

- Assume you have only 1 model
  - Marginalised kernel give you little information: only one feature: P(x|m)
  - To exploit much, the model must be “flexible”, so that we can measure how it adapts to individual items → we require a “smoothly” parametrised model
  - Link with previous approach: locally perturbed models constitute our family of models, but dimF=number of parameters
- More formally, let $P(x|\theta_0)$ be the generative model (θ_0 is typically found by max likelihood); the gradient reflects how the model will be changed to accommodate the new point x (NB. In practice the loglikelihood is used) $\nabla_{\grave{e}} \log P(x \mid \grave{\mathbf{e}})\big|_{\grave{\mathbf{e}}=\grave{\mathbf{e}}_0}$

Fisher Kernel : formally

- Two objects are similar if they require similar adaptation of the parameters or, in other words, if they stretch the model in the same direction:

K(x,y)=

$$(\nabla_{\grave{e}} \log P(x \mid \grave{e})\big|_{\grave{e}=\grave{e}_0})' I_M^{-1} (\nabla_{\grave{e}} \log P(y \mid \grave{e})\big|_{\grave{e}=\grave{e}_0})$$

Where I_M is the Fisher Information Matrix

$$I_M = E[(\nabla_{\grave{e}} \log P(x \mid \grave{e})\big|_{\grave{e}=\grave{e}_0})(\nabla_{\grave{e}} \log P(x \mid \grave{e})\big|_{\grave{e}=\grave{e}_0})']$$

Example 2 : PLSA-Fisher Kernels

- An example : Fisher kernel for PLSA improves the standard BOW kernel

$$K(d_1,d_2) = \sum_c \frac{P(c \mid d_1).P(c \mid d_2)}{P(c)} + \sum_w \tilde{t}f(w,d_1)\tilde{t}f(w,d_2)\sum_c \frac{P(c \mid d_1,w).P(c \mid d_2,w)}{P(w \mid c)}$$

  - where k_1(d1,d2) is a measure of how much d1 and d2 share the same latent concepts (synonymy is taken into account)
  - where k_2(d1,d2) is the traditional inner product of common term frequencies, but weighted by the degree to which these terms belong to the same latent concept (polysemy is taken into account)

“New” perspectives

- Multi-lingual
- Multi-media
- Emotion mining
- Structured documents
- Help to labelling – Active learning