Big Data : Text Mining

Alexandre Saidi

LIRIS-ECL
Février 2018

ECL - Déep. MI

Rappel des objectifs TM

Introduction: contexte, motivation, définitions, applications

Fondements: Bagages théoriques dans
"Informatique Linguistique" (*"Computational Linguistics"*)

Technologie: pour créer un système de "Text Mining"

Applications:
- Résumé automatique,
- Extraction d'information en Biologie,
- Systèmes Q/A (question-answering),
- *Opinion mining*,
- *Sentiment Analysis* (reviews, opinions, votes)
- *Systèmes de Recommandation*
- ... WebMining

Contexte général

Constatation : pour placer des pubs contextuels

Google / Apple / Facebook / Amazon / Ebay / ...

utilisent des algorithmes pour

- épier nos habitudes, usages,
- achats,
- questions posées et les mots choisis
- nos réactions aux réponses aux réponses données
- etc.

Pourquoi ?

- Par amour
- Par philanthropie
- Pour nous espionner (pas faux mais pas de complotisme please !)
- Par intérêt (commerciaux souvent), etc..

Contexte général (suite)

Quelques exemples :

- **Google** détient 67% du marché de recherche d'information sur internet
  - 18% pour microsoft *Bing*, *Yahoo* 11%,
  - le reste pour *Ask*, *AOL*, etc.
- **Facebook** Pré-sélectionne (pour nous) les articles choisis par (l'algorithme de) Facebook (sauf si vos préférences demandent de tout afficher dans l'ordre chronologique).
- **NSA** Data Collection, Interpretation, and Encryption
- **CRUSH** d'IBM (*Criminal Reduction Utilizing Statistical History*) aurait permis à la police de *Memphis* de réduire les crimes de 30% depuis 2006 (chiffres publiés).
- ...

Contexte général (suite)

Le règne du Data : savez-vous qu'en 1 sec , il y (eu) :

- 200000 SMS sur *Whatsapp*
- 400 heurs de séquence *Skype*
- 40000 requêtes *Google*
- 6000 "likes" sur *Facebook*
- Et les biologistes ont crée et stocké leur données sur des supports de stockage à la même vitesse (plus de 29000 Go/s) que celle de la fabrication de ces mêmes supports !

☞ **Origines** : d'où viennent ces données ?

- de nous mêmes (mails, SMS, MMS, Facebook, requêtes Google, nos achats , nos déplacements et GPS, etc...),
- la "communauté" dans laquelle on évolue, ...

• Progrès en moyens de stockage (facteur de ??), rapidité du Transfert et de communication (fibre optique : facteur de 1000), vitesse du traitement (comparez *Motorola 6800* et *Intel I9*)

Contexte général (suite)

Big-Data Textuel : Contexte

- Pléthore de ressources "écrites" :
  livres, journaux, articles, chats, les réseaux sociaux, blogs,...
- On a l'habitude de les exploiter via :
  - les BDs (requêtes) ou
  - l'Extraction d'Information
    - ➛ IR : *Information Retrieval* (cf. Google et équivalents)
- On estime à 80% la part du texte dans ces BDs.
  - ➛ Mais sous exploité !

. . . ⇝

Outils

Outils :

- On distingue (domaine *Pattern Recognition* , *KD*):
  - Information Retrieval (IR)
  - Knowledge Extraction (eg. IE/KE)
    - ☞ Machine Learning & Data Mining
  - Text Mining (TM)

Text Mining (ou TDM = KDT) :

- Analyse intelligente de texte
- **Objectif** (principal) : extraction de connaissances (et d'information non triviale) depuis un corpus de texte non-structuré.

Outils (suite)

Qu'est-ce qu'on fouille / cherche :

- Dans les mails, MSN, Blogs, on cherche
  - Des entités (personnes, compagnies, organisations, ...)
  - Des evts : inventions, offres, annonces, attaques, ...
- Dans les articles scientifiques, livres :
  - On cherche des sujets / tendances

    ☞ En bio-Info/bio-math, le vocabulaire spécifique est un pb. !
- Dans les journaux : on peut avoir besoin d'un résumé, ...
- En Génie Logiciel (les docs) :
  - les capacités / perf. à partir de la spécification,
  - des conflits entre le code source et la documentation, · · · ⇝

Outils (suite)

- Sur le Web
  - Chercher de nouveaux produits dans les pages Web des compagnies (pour faire beaucoup de $$$)
  - Chercher du contenu illicite !
  - Trend Mining, Opinion Mining, autres *Novelty Detection*
  - Création d'ontologie, Entity Tracking, IR, ...

  ➛ Sur ce point, une difficulté vient du multi-média + hyper liens + infos cachées ("deep/dark web")

- Détection de Spam
- Classification : ordre / offre
- Clustering de grosses corpora (*vivisimo / IBM*)
- Topic maps (*leximancer.com*)

☞ Le plus gros système existant : ECHELON (UK/USA)

➛ Fouilles industrielles / politique-stratégique / ...

Outils (suite)

- En TLN (traitement de langues Nat. = NLP) **classique** :
  - Traduction Automatique (Machine Translation)
  - Résumé automatique (Automatic summarization)
  - QA
  - Correction de texte (Error handling)
- Problème++ si analyse multi-langues et cross-document. ...
  - ➛ Sans parler du "temps" (analyse temporelle)

Approche moins classique / plus récente :

on voudrait un assistant intelligent (humain ?) qui extrait, traite, condense et résume l'info (Knowledge) pour nous

- ➛ + visualisation agréable si possible !

- On peut faire l'analyse de réponses "libres" dans les sondages
- L'analyse des contrats d'assurance / garantis / diagnostics, ...
- Le traitement automatique de mails, messages, chats, ..

☞ On dispose de IR (e.g. Google) mais c'est du Data ($\neq$ Knowledge)

- ➛ il faudrait traiter les réponses de Google (cf. Q/A) ...

Qu'est-ce que Text Mining

Qu'est-ce que Text Mining ?

> *La découverte par l'ordinateur (ML) de nouvelles informations (Knowledge) méconnues en extrayant ces informations d'une (grande) quantité de corpora textuels non structurés.*

☞ Information précédemment méconnue ?

- Découverte de vraies nouvelles informations (infos inattendues)
  - ➛ Ce n'est pas la découverte des motifs simples

> Comme la différence entre un détective qui suit des pistes pour trouver un criminel **vs**. un analyste qui observe des statistiques criminelles pour découvrir des tendances par exemple en vol de voiture.

Qu'est-ce qu'un corpus non structuré ?

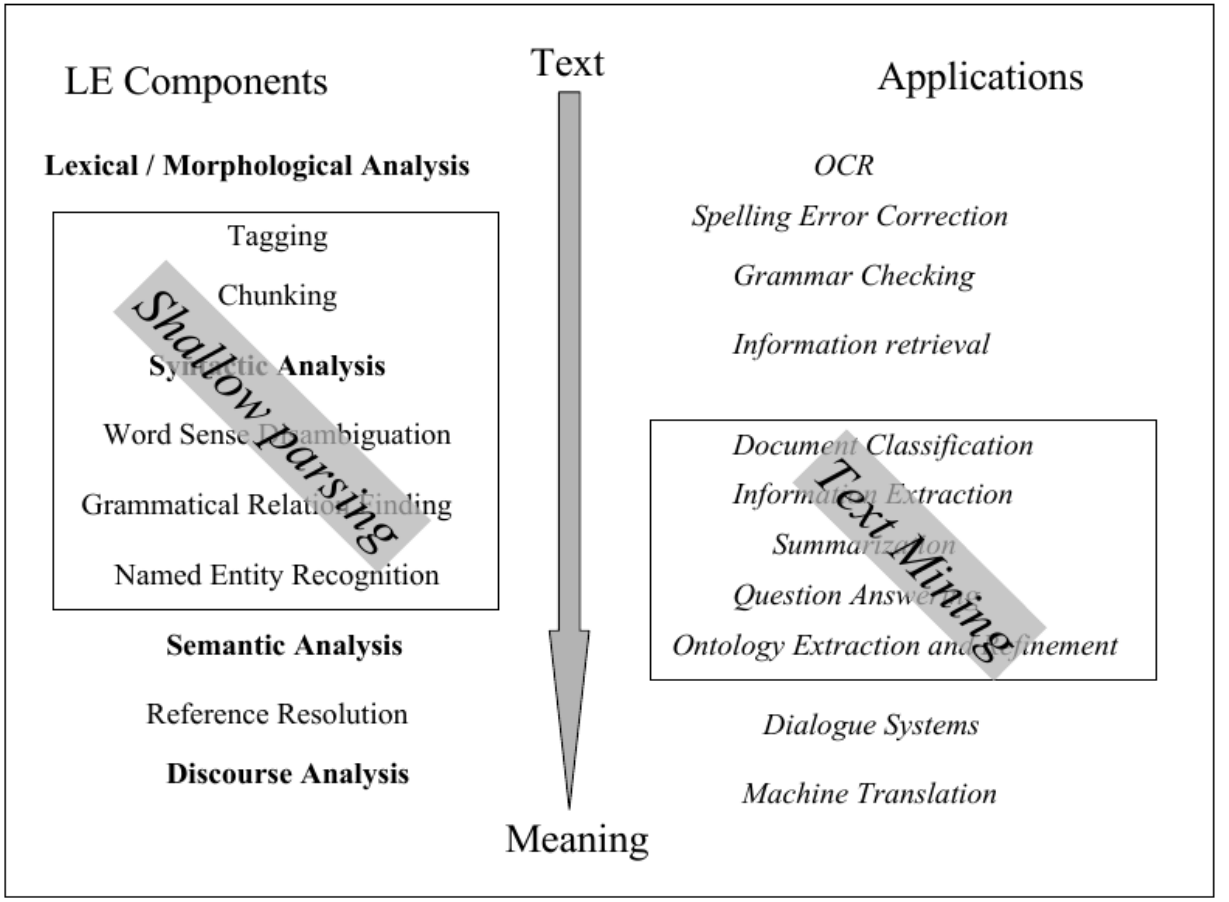
- Texte libre (syntaxiquement)
- Par opposition à une phrase juste ou du HTML/XML (avec balises)

...

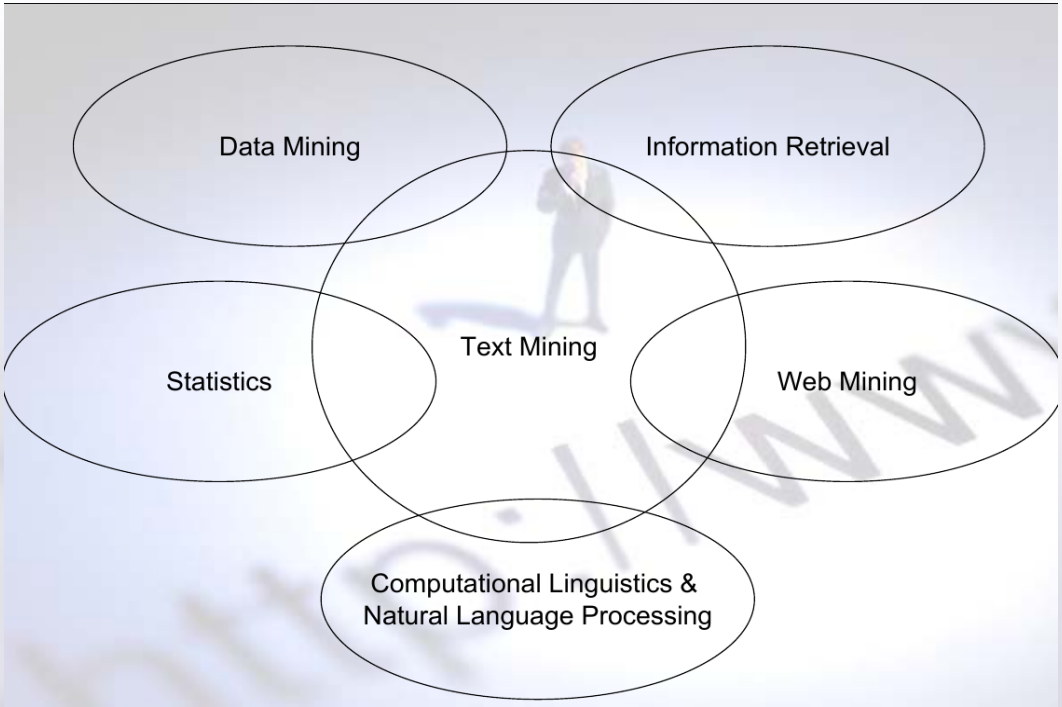
☞ Machine Learning sur les Images est assez proche du Text Mining.

Qu'est-ce que Text Mining (suite)

Linguistic Component Extraction (LE) vs. Text Mining



Qu'est-ce que Text Mining (suite)

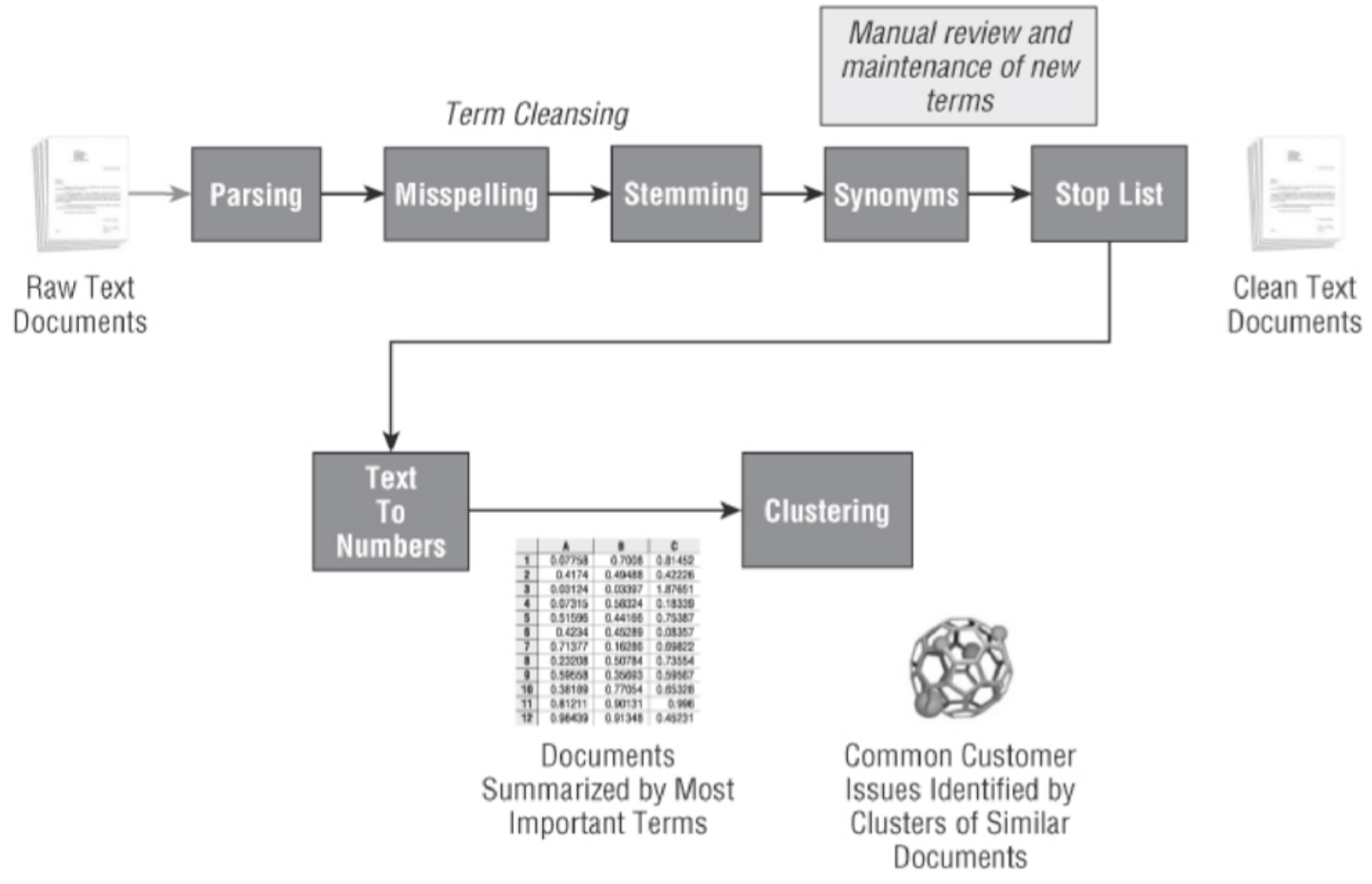


- TM traite du texte libre en langue naturelle.
- Vient du domaine "Computational Linguistics" (CL) et NLP dont des applis pratiques et concrètes sont appelées *Language Technology* (**LT**).

. . . ⇝

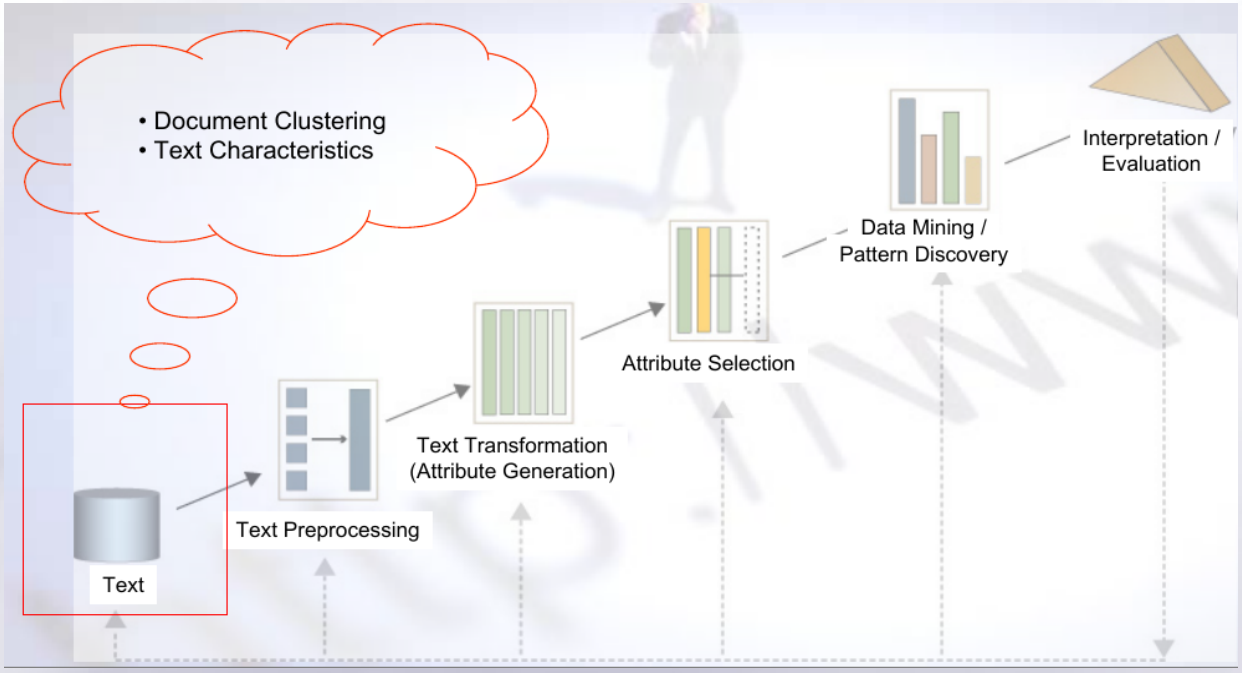
Traitements

Un scénario courant de Text Mining :



Traitements (suite)

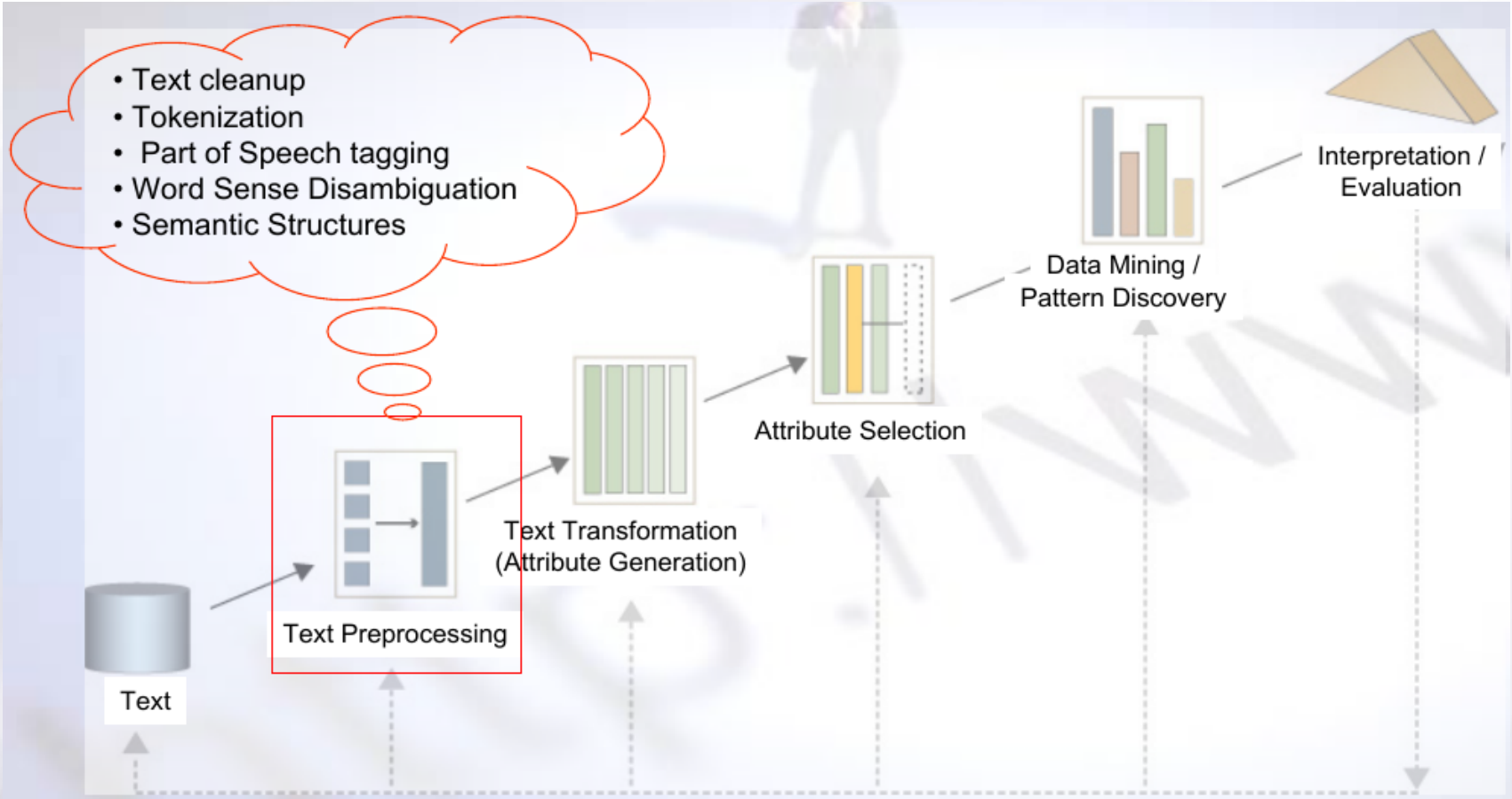
Divers Objectifs des traitements



Que cherche-t-on à faire ?

- Faire un cycle Text Mining / WenMining/ Analyse / ...
- Ou par exemple faire un Clustering (classification non supervisée) pour choisir les documents "relevants".
- Etc.

Pré processing



Pré processing (suite)

- **Nettoyage**
  - Faire du OCR si nécessaire (avec ses erreurs)
  - Enlever pubs, barre de navigation, etc. (des pages WEB)
  - Normaliser, convertir .doc / .pdf / données binaires / ...
  - Traiter tables, formules, légendes, ... et les figures ?

- **Tokenisation** : isoler des mots
  - Certains langues (cf. chinoise) n'ont pas d'espace :
    ➛ il faut segmenter les séquences ?
  - La langue allemande a des déclinaisons trop compliquées
    ➛ il faut décomposer...
  - Certains domaines (médecine, Bio, Chimie) ont des termes et notation "étranges" (vous savez lire une ordonnance ?).

Pré processing (suite)

Le traitement

- (Après la) Suppression des ponctuations
- Suppression des mots usuels
  Les mots tels que "le", "la", "tu", "vous", ... (en anglais "the", "a", "an",...) n'apportent a priori pas d'information intéressante.
- **Stemming** ("racinisation", "désuffixation")
  - Identifier le radical / la racine ("stem" = souche) des mots
  - Réduire la dimension (le nombre de termes réduit, cf. SVD)
    - ➛ Ex : flying/flew deviennent "fly"
- Deux algorithmes utilisés : *Porter* et *KSTEM*
- Exemple de *stemming* par les deux algorithmes : ... ⇝

Pré processing (suite)

- Le texte original :

"Document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals."

- Résultats **Porter** (après suppression des ponctuations)

"market strateg carr compan agricultur chemic report predict market share chemic report market statist agrochem"

- Résultats **KSTEM**

"marketing strategy carry company agriculture chemical report prediction market share chemical report market statistic"

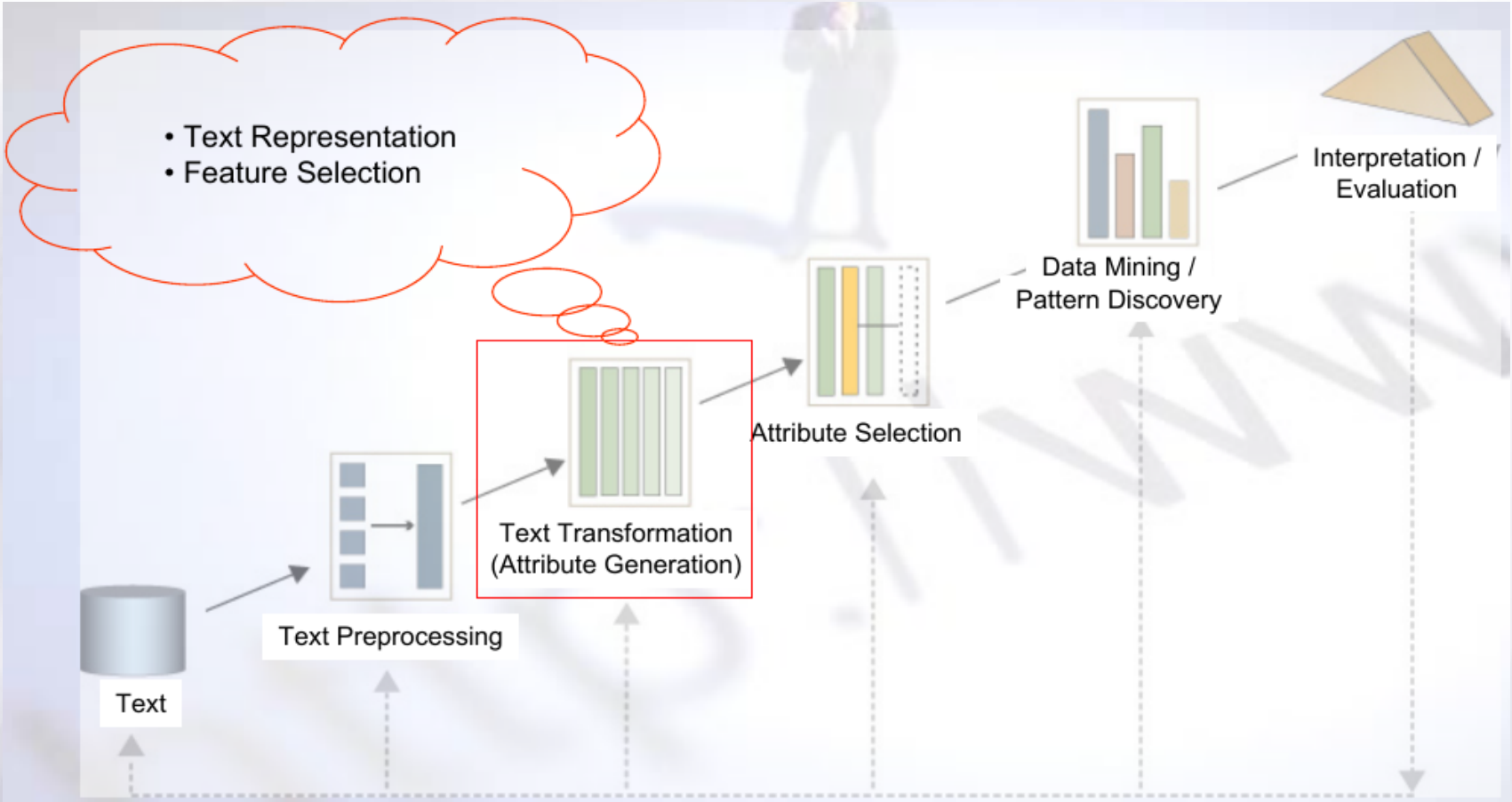
Pré processing (suite)

Alternative au "Stemming" : **Lemmatisation**

- **Lemmatisation** : trouve une racine (*Lemme*) $\in$ dico
  - Nécessite un POS (part of speech analysis)
    - ➛ POS : extraction de la catégorie lexicale
  - Lemmatisation peut nécessiter une analyse **Morphologique**
- L'analyse Morphologique faite sur le sens des parties du mot :
  - le suffixe "eur" n'a pas toujours le même sens
    - → dans "chanteur" = une personne,
    - → dans "écailleur" = un objet pour écailler
  - "arm" est distingué de "army" (pas en Stemming)
- Pour TM, il faudrait plutôt la "Lemmatisation" (mais le cout ?)
- Mieux : Lemmatisation à base de ML
  on apprend des règles puis on utilise les résultats

Génération des attributs

Génération des attributs



Génération des attributs (suite)

Techniques de génération :

- *Concept Chunking*
  - Identification de concepts e.g. (noms de) "maladies", "lieu" (d'une conf, d'une catastrophe, ...), une "date" (soumission d'un article), etc...
  - Par ex. dans les Annonces (p. ex. de séminaires), en biologie moléculaire, News, rapports d'accidents, ...
  - Les systèmes Q/R utilisent souvent du *Concept Chunking*.
  - Le *Concept Chunking* ne prend pas en compte le contexte du document : trop lourd !
- Annotation *part-of-speech* (POS)
  - Étiquetage des mots dans un texte par leur label de POS (leur classe lex/synt.)
    - ➛ Ex. : Noun, Preposition, Conjonction, Pronoun, Verb, Adverb, Adjectif, . . .
  - A base de règles grammaticales et/ou
  - A base de statistiques :
    - ➛ S'appuie sur la proba d'apparition de mots
    - ➛ Ou bien on apprend (ML) à annoter le corpus

Génération des attributs (suite)

- *Named Entity Recognition* (**NER**) :
  - Une combinaison de *concept chunking* + labeling de ces chunks,
  - Identifier des informations textuelles : les "personnes", "lieux", "organisations", "compagnies", "objets", etc...
  - Utilise un Tokenizer + POS au lieu de faire une Analyse syntaxique.
- *Désambiguïsation* du sens (sémantique) des mots
  - Trouver le sens d'un mot utilisé dans une phrase
    - ➛ "Les poules du couvent couvent"
    - ➛ "L'ami de mon oncle qui habite Paris"
    - ➛ "The king saw the rabbit with his glasses"
      - → Combien de sens ?

Syntaxe

- A ce stade, chaque phrase est devenue une succession de *Tokens* : de *Lemmes* (*Stem* moins intéressant ici)
- **Dans certains cas**, on procède à un *POS tagging* (POS : part-of-speech) :

  Ex. : *"Les poules du couvent couvent"* (avec tags multiples, homonymes)

  - Les tags : nom, adj, verbe, adv, pronom, det, préposition, ...
    - ➛ De 45 à 146 tags suivant l'outil
  - On peut faire un *apprentissage* pour pouvoir taguer
    - ➛ Méthodes DM pour taguer : D. Trees, HMM, SVM, ...
    - ➛ D'autres techniques existent (propres à TLN)
- Au besoin, on peut ensuite appliquer une Analyse Syntaxique :
  - ➛ On a "Souché" (Stem) / Lemmatisé : analyse synt. de "surface" ?
  - ➛ Donne un arbre syntaxique
  - ➛ Traitement **compliqué** !
  - ➛ Les textes libres : s'abstenir !

Syntaxe (suite)

- Exemple de structure de *Chunk*

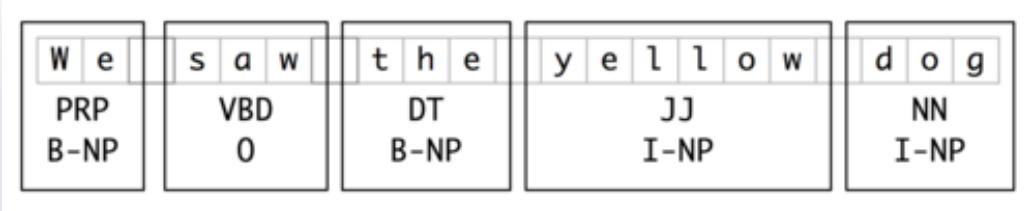
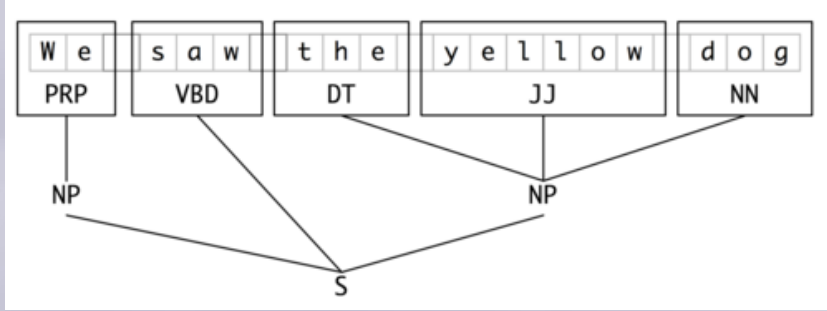


Figure 1: **I** (inside), **O** (outside), or **B** (begin)

- Exemple de structure de Syntaxique (arbre)



☞ Essayer http://nlp.stanford.edu:8080/parser/index.jsp

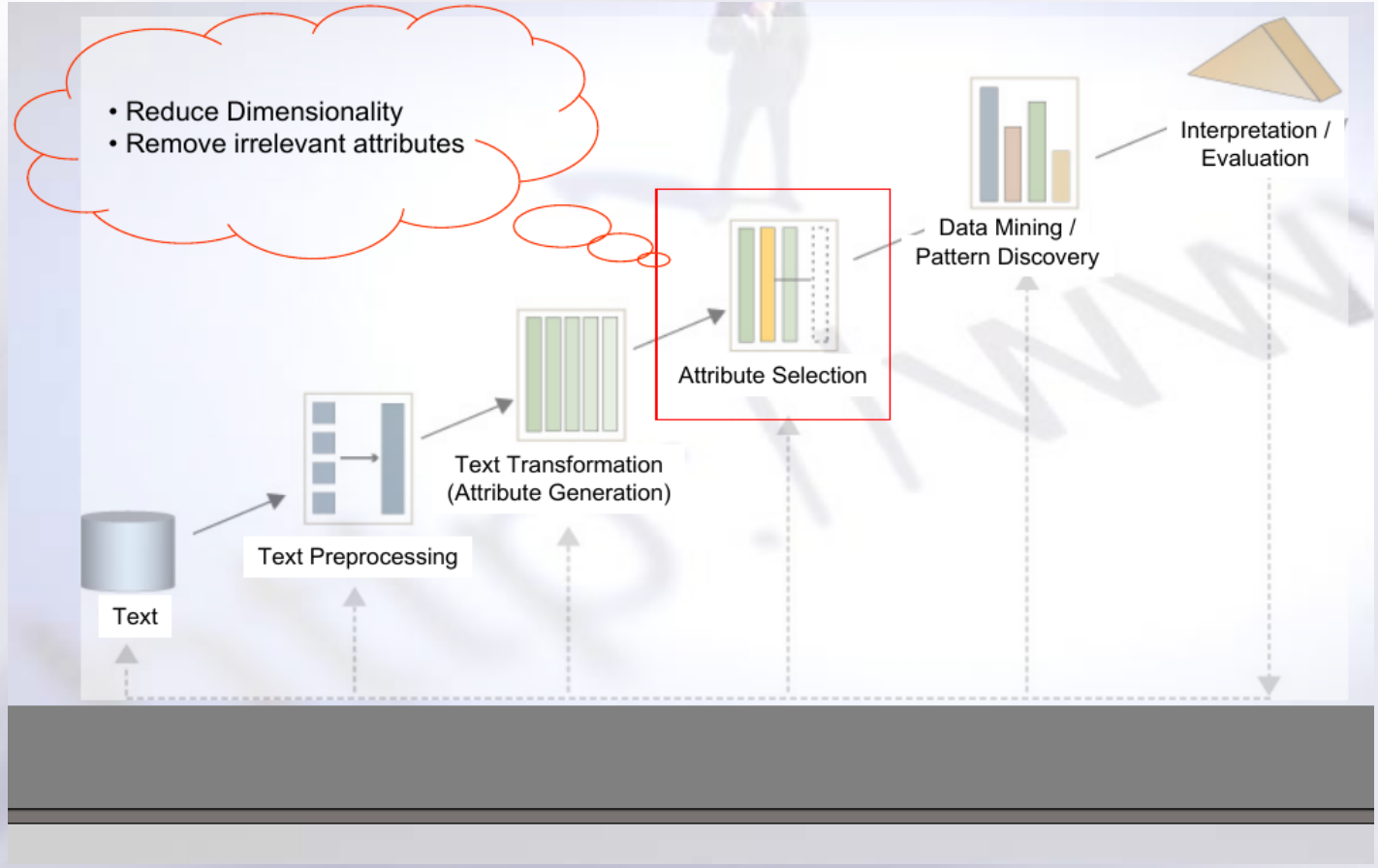
Bilan provisoire

- **Structures** : 2 méthodes
  - Analyse **complète** : produit un arbre syntaxique pour une phrase
  - Analyse **partielle** : produit des constructions syntaxiques (*syntagmes*) telles que "groupe nominal" ou "groupe verbal"

- Que choisir ?
  - Un arbre complet syntaxique est très difficile à construire :
    - ➛ le texte est "libre",
    - ➛ peut contenir des nouveaux mots,
    - ➛ erreurs de frappes et d'écriture,
    - ➛ erreurs dans le processus POS, ...
  - Question de coût :
    - ➛ utiliser plutôt les structures syntaxiques partielles.

sélection d'attributs

Vers une BD : sélection d'attributs



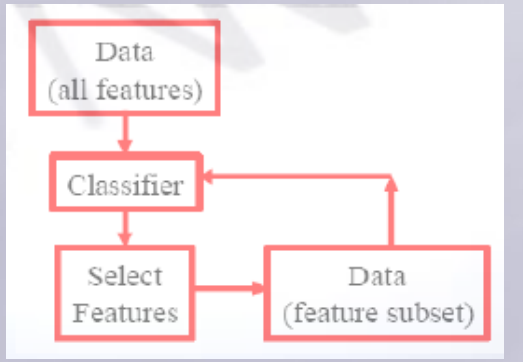
sélection d'attributs (suite)

- Les techniques précédentes permettent la génération d'attributs.
- Après l'étape de génération des attributs (cf. NE),
  Un document sera représenté par ses **termes** (*features*) et leurs occurrences (+ tags éventuels).
- Choix des attributs (**feature selection**) :
  ➛ Quels attributs représentent le mieux un document ?
- Il y a des techniques pour générer des attributs
- On prend en compte des caractéristiques :
  ➛ mot $\rightarrow$ terme $\rightarrow$ attribut (label, catégorie, . . .)
- Puis on sélectionne les attributs qui seront nos **vars explicatives**

sélection d'attributs (suite)

Une méthode employée :

- Sélection d'attributs pour une utilisation dans le classifieur
- On peut intégrer / adapter cette étape au classifieur
  - Le classifieur devient une partie de "feature selection"
  - Processus itératif
  - Approche plus ad-hoc, mieux adaptée
  - Coût plus élevé (il faut entraîner le classifieur)
  - Le classifier doit pouvoir évaluer les features (on lui apprend !)



Sémantique

Selon l'utilisation, on peut (maintenant) passer à la sémantique (ou pas !).

Quelques outils :

- **WordNet**(s)
  Un réseau sémantique qui encode les mots d'une langue via
  - Synsets : le sens de chaque mot (eg. banque)
  - Relations : synonymie, hypernymie (synonymie moins générale), homonymie, hyponymie (Cyan est hyponyme de Bleu), antonymie, holonymie (guidon est holonyme de vélo), ...

  ➛ Un Wordnet anglais encode 147249 mots (v2.1)
  ➛ Un Wordnet français libre : en dév. (un fichier xml).

- Ex : voyons si "thé" (tea) est qq chose qu'on peut boire : ··· ⇝

Sémantique (suite)

WordNet Search - 3.1
- WordNet home page - Glossary - Help

Word to search for: tea Search WordNet

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **tea** (a beverage made by steeping tea leaves in water) *"iced tea is a cooling drink"*
- S: (n) **tea**, afternoon tea, teatime (a light midafternoon meal of tea and sandwiches or cakes) *"an Englishman would interrupt a war to have his afternoon tea"*
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - S: (n) meal, repast (the food served and eaten at one time)
      - ***direct hyponym*** / *full hyponym*
        - S: (n) mess (a meal eaten in a mess hall by service personnel)
        - S: (n) square meal (a substantial and nourishing meal) *"he seldom got three square meals a day"*
        - S: (n) potluck (whatever happens to be available especially when offered to an unexpected guest or when brought by guests and shared by all) *"having arrived unannounced we had to take potluck"; "a potluck supper"*
        - S: (n) refection (a light meal or repast)
        - S: (n) breakfast (the first meal of the day (usually in the morning))
        - S: (n) brunch (combination breakfast and lunch; usually served in late morning)
        - S: (n) lunch, luncheon, tiffin, dejeuner (a midday meal)
        - S: (n) **tea**, afternoon tea, teatime (a light midafternoon meal of tea and sandwiches or cakes) *"an Englishman would interrupt a war to have his afternoon tea"*
        - S: (n) dinner (the main meal of the day served in the evening or at midday) *"dinner will be at 8"; "on Sundays they had a large dinner when they returned from church"*
        - S: (n) supper (a light evening meal; served in early evening if dinner is at midday or served late in the evening at bedtime)
        - S: (n) buffet (a meal set out on a buffet at which guests help themselves)
        - S: (n) picnic (any informal meal eaten outside or on an excursion)
        - S: (n) bite, collation, snack (a light informal meal)
        - S: (n) nosh-up (a large satisfying meal)
        - S: (n) ploughman's lunch (a meal consisting of a sandwich of bread and cheese and a salad)
        - S: (n) banquet, feast, spread (a meal that is well prepared and greatly enjoyed) *"a banquet for the graduating seniors"; "the Thanksgiving feast"; "they put out quite a spread"*
      - *part meronym*
      - *direct hypernym* / *inherited hypernym* / *sister term*
  - *domain region*
- S: (n) **tea**, Camellia sinensis (a tropical evergreen shrub or small tree extensively cultivated in e.g. China and Japan and India; source of tea leaves) *"tea has fragrant white flowers"*
- S: (n) **tea** (a reception or party at which tea is served) *"we met at the Dean's tea for newcomers"*
- S: (n) **tea**, tea leaf (dried leaves of the tea shrub; used to make tea) *"the store shelves held many different kinds of tea"; "they threw the tea into Boston harbor"*

Sémantique (suite)

Pourquoi la sémantique (le sens) est importante ?

• Si on a le sens (correct) de chaque mot, on peut (par exemple) passer à une représentation en logique des prédicats.

- ➛ Le raisonnement logique devient possible.
- ➛ Ex. : en parlant des "compagnies", de la phrase
  "X bought Y" ou "Y was acquired by X",

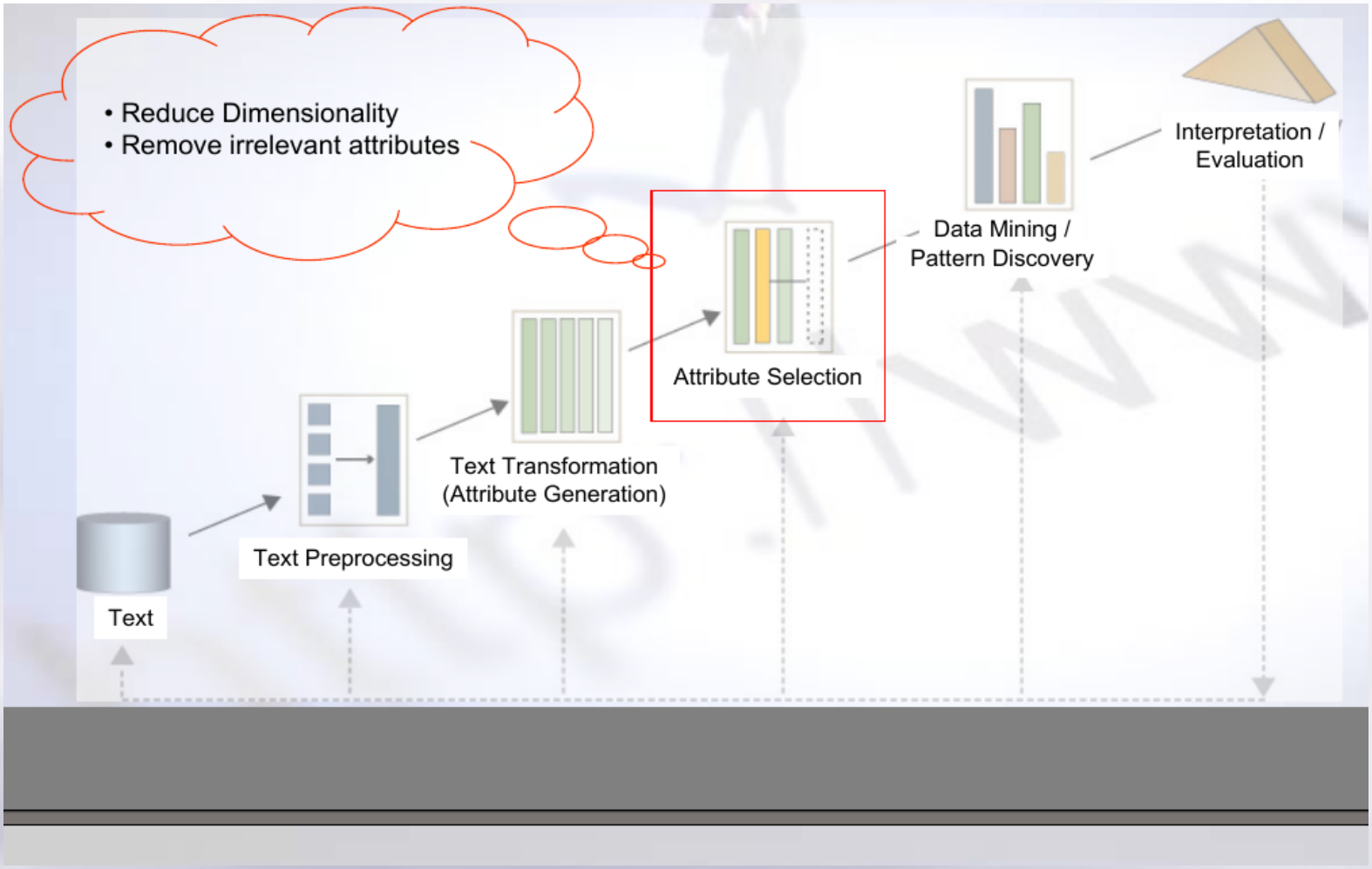
  on peut passer à :
  $$company(X) \wedge company(Y) \wedge buy\_act(X, Y)$$
- ➛ On peut y intégrer les informations sémantiques venant e.g. de Wordnet.
- ➛ On peut mieux trouver des documents pertinents,
- ➛ On peut traduire , ...

Réduction de Dimension

Rappel des étapes du Pré-process

- **Nettoyage du texte :**
  - Suppression des pubs des pages Web,
  - Conversion au format binaire (présence/absence d'un mot dans un doc donnée donnant une matrice de 0/1,
  - Normalisation de ce dernier
  - Traitement des figures, tables et formules, etc.
- **Tokenisation/Stemming**
  - Scinder les chaines de caractères en un ensemble de tokens.
  - On traite les apostrophes : *aujourd'hui* : un mot ou 2 ?
  - Hyphens et coupure : database ou data-base ?
  - Cas spéciaux : "C++", "A/D", ":-)", "..." ?
  - Espaces : parfois plusieurs espaces peuvent avoir un sens ..
- **POS tagging / Chunking / Lemmatisation / ...**
- **Génération et sélection d'attributs**

Réduction de Dimension (suite)



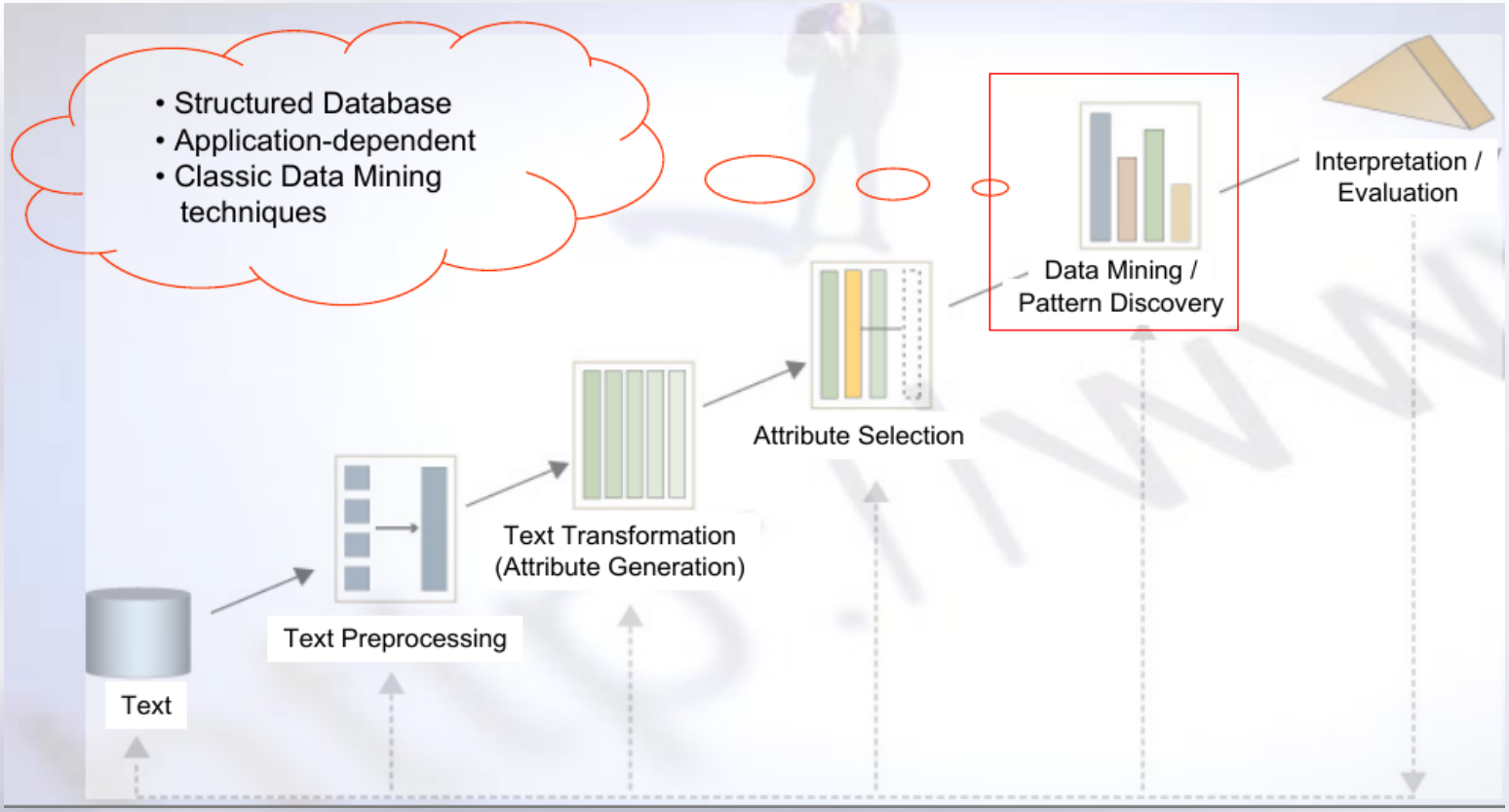
Réduction de Dimension (suite)

Réduction de dimension :

- La grande dimension (beaucoup de termes) pose problème
- Question de ressources (temps/espace) de calcul limitées,
  - ➛ La "faisabilité" dépend de la taille du problème
- Certains attributs n'apportent pas beaucoup d'info
  - ➛ P. ex. un nom (lexème) dans un article d'actualité n'aidera pas à la classification de "politique" ou "sport".
- Outils : ACP / SVD / Factorisation de matrices / ...

Phase de Mining

DM : Text Mining



Phase de Mining (suite)

- Arrivé à ce stade, la suite coïncide avec les techniques DM "traditionnelles".
- Ces techniques sont utilisées sur les BD structurées issues des étapes précédentes.

☞ Nécessite l'utilisation des techniques d'Extraction de Connaissances.

  - Clustering
  - Classification
  - etc.

- Aux côtes des traitements abordés, on peut utiliser des techniques (plus simples) de IR.
  - ➛ Utilisées en *clustering* (simple) de documents

Quelques méthodes de DM

Clustering de Documents

- Grande Volume de données textuelles
  Milliards de documents à traiter efficacement.
- On ne sait pas d'avance quels sont les documents intéressants (à notre recherche)
- Clustering : regrouper les documents *similaires*
  ➛ Par ex, la classe des (journaux) d'info, de sports, de spam, ...
- Une solution utilisée : *clustering* de documents
  ➛ (un Apprentissage Non-supervisé)
- Les méthodes (simples) de *clustering* les plus utilisées
  - K-means (voir ci-après)
  - Bayésiennes (un exemple ci-après)
  - Clustering Hiérarchique agglomératif (top-down).
  - ...

Quelques méthodes de DM (suite)

K-mens
Algorithme :

- Étant donné
  - Un ens. de doc. (présentés e.g. par une repr. vectorielle)
  - Une métrique de distance (e.g. cosinus)
  - K clusters à trouver (K est un param)
- Initialiser k groupes avec un centroïde (médoïde)
  - ➛ P. Ex. un doc. choisi aléatoirement (du corpus)
- Tant que non convergé
  - Attacher chaque document au centroïde le plus proche
  - Re-calculer le centroïde (Médoïde) dans chaque groupes

☞ Une variante de la méthode EM.

Quelques méthodes de DM (suite)

Bayes Naïve en classification de docs

• Soit un corpus où chaque doc représente une instance d'une classe (*Topic*) de documents (cf. issue d'un clustering)

➛ Par ex, la classe des (journaux) d'info, de sports, de spam, ...

• Les docs sont caractérisés par les termes qui les constituent.

• Les fréquences des mots prises en compte en appliquant une forme modifiée de Bayes naïve appelée *multi-nominal Naïve Bayes* (MNB).

➛ Dans **MNB**, les docs sont considérés comme des *sacs-de-mots*

➛ Un doc : un ensemble pouvant contenir plusieurs fois le même mot.

• **Deux méthodes basiques :**

1- Une méthode naïve mais plus rapide : traiter la présence/absence d'un mot, puis de décider sur ces simples infos;

2- Mieux : tenir compte du nombre d'occurrences d'un mot qui peut être importante pour la classification (cf. *fréquence*, `tfidf`) ../..

Quelques méthodes de DM (suite)

- MNB s'appuie sur une distribution *multinomiale* pour la classification.
- Pour cette distribution, la probabilité d'un doc E (composé de n_E mots clefs, voir ci-dessous) soit d'une classe H est :

$$\Pr[H|E] \propto \Pr(H) \prod_{j=1}^{n_E} \Pr[w_j|H]$$

  - w_j : les mots (clefs du dico) des documents de la catégorie H
  - $\Pr[w_j|H]$: la proba que w_j figure dans les docs de la catégorie H
    ➛ combien w_j contribue à ce que H soit la bonne classe de E.
  - $\Pr(H)$: probabilité *a priori* qu'un doc soit de la catégorie H.
- Le but est de trouver la meilleure classe H pour E,
  ➛ i.e., celle la plus vraisemblable
  ➛ celle avec une probabilité *a posteriori* maximum
  ➛ (*MAP = maximum a posteriori*).

☞ Ceux qui ont fait du DM : les chanceux !

Quelques méthodes de DM (suite)

- $w_1, ...w_j...w_{n_E}$ sont les mots clefs dans la doc E
  avec n_E = nombre de ces mots dans E
  - ➛ P. ex, $w_1, ...w_j...w_{n_E}$ pour un doc E avec une seule phrase :

    "ECL et EML sont ensemble sur un Bateau"

    sera <*ECL, EML, ensemble, Bateau*> avec n_E=4 (les *stems*)
- On choisira le maximum du log-vraisemblance :

$$\log(\hat{Pr}[H|E]) \propto \log(\hat{Pr}(H)) + \sum_{j=1}^{n_E} \log(\hat{Pr}[w_j|H])$$

  - Cette somme indique "combien" E peut être de la classe H.
  - $\log(\hat{Pr}[w_j|H])$: la val. de l'indicateur w_j pour désigner la classe H
  - $\log(\hat{Pr}(H))$: la fréquence relative de la classe H :
    - ➛ plus la classe est fréquente, plus elle a la chance d'être choisie.

N.B. on choisit $\hat{Pr}$: une estimation de Pr car les vraies $Pr(H)$ et $Pr[w_j|H]$ sont inconnues mais on fait une estimation à partir d'une base d'apprentissage. ... ⇝

Quelques méthodes de DM (suite)

Comment estimer $\hat{\Pr}(H)$ et $\hat{\Pr}[w|H]$?

• On utilisera MLE qui est ici simplement la fréquence relative dans la base d'apprentissage :

• $\hat{\Pr}(H) = \frac{N_H}{N}$ où N_H est le nombre de documents dans la classe H et N est le nombre total des documents du corpus.

• L'estimation pour $\hat{\Pr}[w_j|H]$ est la fréquence relative du mot w_j dans les documents de la classe H.

$$\rightarrow \hat{\Pr}[w|H] = \frac{T_{Hw}}{\sum_{w' \in V} T_{Hw'}}$$

➛ T_{Hw} est le total de toutes les occurrences du mot w dans la base d'apprentissage (dans le vocabulaire des *mots clefs* V).

Quelques méthodes de DM (suite)

Rappel : la position d'un mot dans les docs n'est pas exploitée (pas d'ordre)

- On ne calcule pas différentes estimations pour différentes positions
  ➛ Si un mot apparaît 2 fois, alors $\hat{\Pr}[w|H]$ sera identique pour les 2.
- Ex. : les docs *{Ecully Dardilly Ecully}* et *{Ecully Ecully Dardilly}* sont considérés identiques et les mots répétés ont le même poids.

- Le pb. du *véto du zéro* : on a recours au *Lissage de Laplace* :

$$\hat{\Pr}[w|H] = \frac{T_{Hw} + 1}{\sum_{w' \in V}(T_{Hw'} + 1)} = \frac{T_{Hw} + 1}{(\sum_{w' \in V} T_{Hw'}) + B}$$

où B est la constante du lissage de Laplace
➛ Ici, B= la taille du vocabulaire.
➛ L'ajout de 1 signifie une *a priori* uniforme (comme si chaque mot apparaissait une seule fois dans chaque classe).

(Voir Bayes pour la généralisation du lissage.)

Quelques méthodes de DM (suite)

Un exemple : soit les documents

ID	Ensemble	les mots du document	classe H="Chine" ?
1	Apprentissage	Chinois Pékin Chinois	oui
2	Apprentissage	Chinois Chinois Shanghai	oui
3	Apprentissage	Chinois Macao	oui
4	Apprentissage	Tokyo Japon Chinois	no
5	Test	Chinois Chinois Chinois Tokyo Japon	?

- On a également, pour $\mathrm{H} = \text{"Chine"}$, $\hat{\Pr}(\mathrm{H}) = 0,75$ et $\hat{\Pr}(\mathrm{H}) = 0,25$
- Calcul des probabilités conditionnelles :

$$\hat{\Pr}(\text{"Chinois"}|\mathrm{H}) = \frac{5+1}{8+6} = \frac{3}{7}$$

$$\hat{\Pr}(\text{"Tokyo"}|\mathrm{H}) = \hat{\Pr}(\text{"Japon"}|\mathrm{H}) = \frac{0+1}{8+6} = \frac{1}{14}$$

$$\hat{\Pr}(\text{"Chinois"}|\mathrm{H}) = \frac{1+1}{3+6} = \frac{2}{9}$$

$$\hat{\Pr}(\text{"Tokyo"}|\mathrm{H}) = \hat{\Pr}(\text{"Japon"}|\mathrm{H}) = \frac{1+1}{3+6} = \frac{2}{9}$$

➛ Les dénominateurs $(8+6)$ et $(3+6)$: car la longueur des textes de la classe $\mathrm{H} = \text{"Chine"} = 8$ et celle de $\overline{\mathrm{H}} = 3$ et

➛ La cste B du lissage de Laplace = 6 (= nb. termes dans le vocab.)

Quelques méthodes de DM (suite)

- On aura $\hat{\Pr}(\text{"Chine"}|d_5) \propto 3/4.(3/7)^3.1/14.1/14 \approx 0,0003$
  Et $\hat{\Pr}(\overline{\text{"Chine"}}|d_5) \propto 1/4.(2/9)^3.2/9.2/9 \approx 0,0001$

- NB : pour passer de $\approx$ à $=$, on normalise en divisant ces valeurs par leur somme :

$$\hat{\Pr}(\text{"Chine"}|d_5) = \frac{0,0003}{0,0003 + 0,0001} = 75\%$$

Et $$\hat{\Pr}(\overline{\text{"Chine"}}|d_5) = \frac{0,0001}{0,0003 + 0,0001} = 25\%$$

Conclusion :

le document de test (d_5) appartient à la classe $H = \text{"Chine"}$.

➛ Ici, les 3 occ de l'indicateur positif ("Chinois") dans d_5 prennent le dessus sur les 2 occ des indicateurs négatifs ("Japon" et "Tokyo").

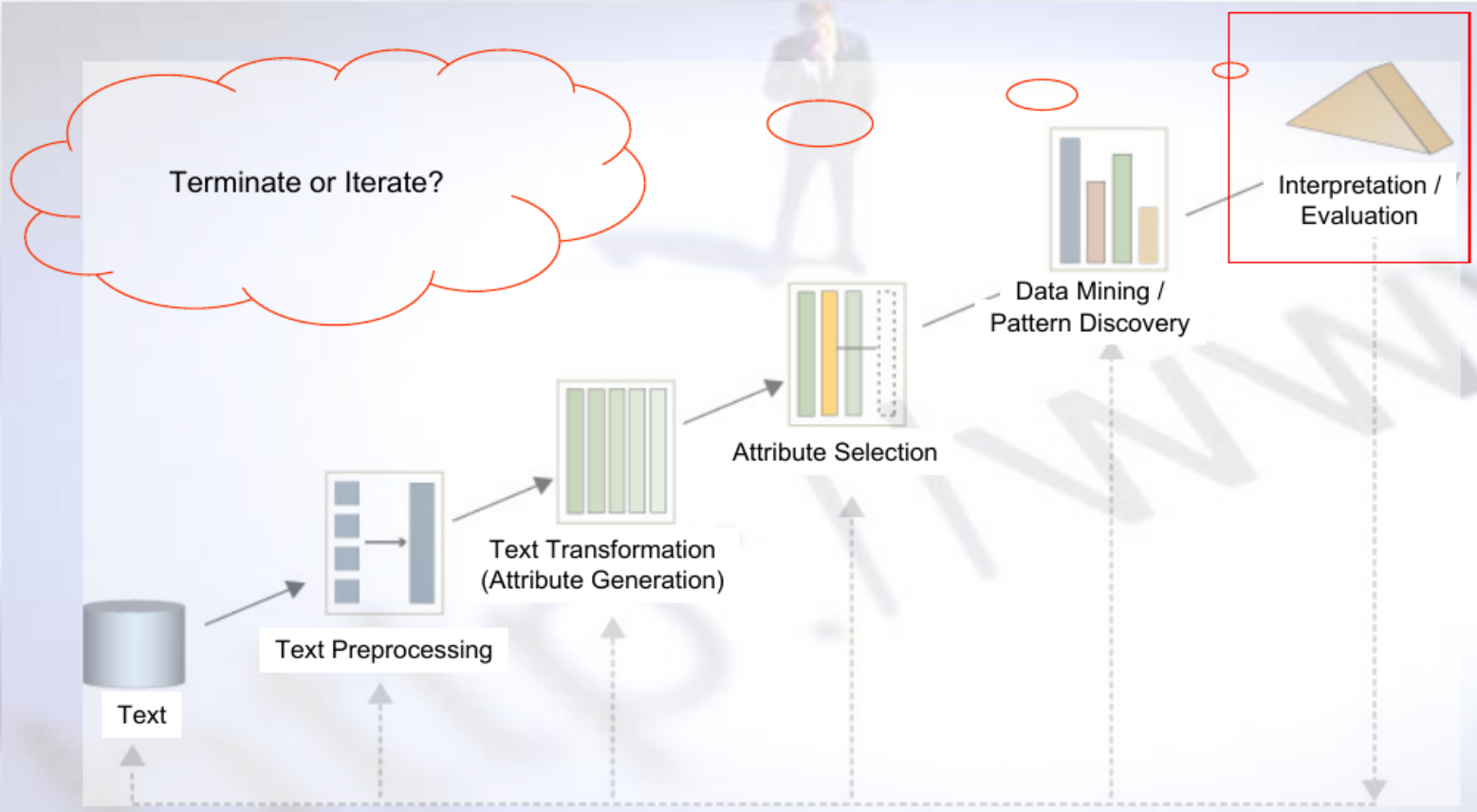
☞ L'hypothèse de l'indépendance des mots dans la phrase !

➛ Une réponse : utilisation de Digrammes, Trigrammes, ...

evaluation

Evaluation (simple)

☞ Tout apprentissage artificiel doit être validé.



evaluation (suite)

Évaluation : quelques mesures de base (dans IR)

- Appliqué aux réponses :

$$\text{Precision} = \frac{\text{correcte} + 0.5 * \text{partiel}}{\text{correcte} + 0.5 * \text{partiel} + \text{incorrect}}$$

$$\text{Recall} = \frac{\text{correcte} + 0.5 * \text{partiel}}{\text{correcte} + 0.5 * \text{partiel} + \text{manquants}}$$

Et en utilisant les 2 précédentes mesures ($\beta = 1$ si même poids)

$$\text{F} - \text{mesure} = \frac{(\beta + 1)^2.\text{P}.\text{R}}{\beta^2.\text{R} + \text{P}}$$

- D'autres mesures existent.
- Le rôle de l'expert (si disponible).

Addendum : Les outils et la technologie existants

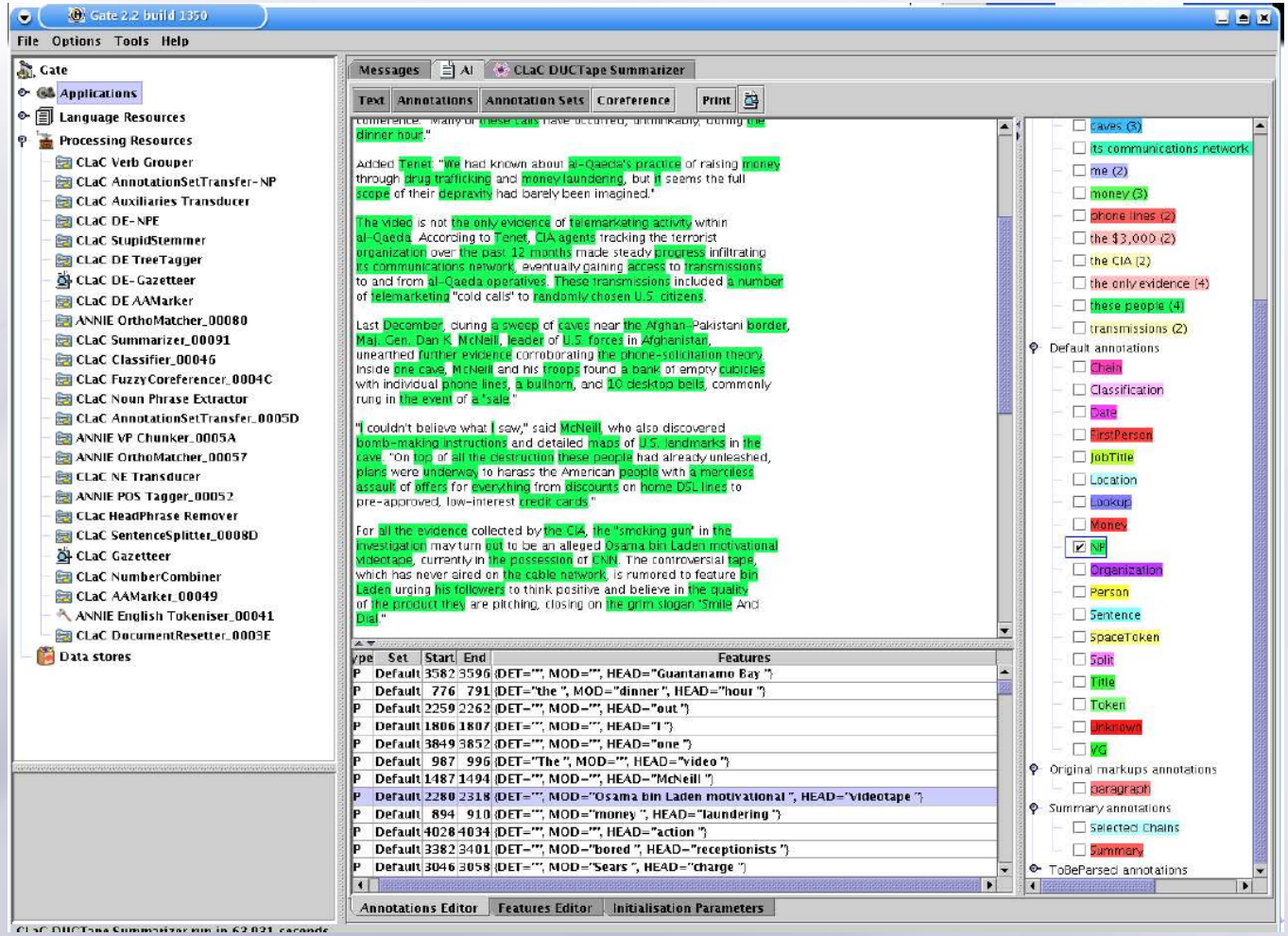
- Un outil TM doit disposer de
  - Outils de traitement de documents (divers formats à convertir)
  - Outils d'annotation
  - Traitements : Tokenisation, Lemmatisation, chunking, POS tagger, AEF, Parsers, Classifiers, Entity taging, Summarisation, ...
  - Wordnets, Lexicons
  - Grammaire et modèles de langues
  - ...

Exemples :

- Gate : General Architecture of Text Engineering
- UIMA : Unstructured Information Management Architecture (IBM)
- ANNIE : outil d'extraction d'information ($\neq$ google)
  propose un outil d'extraction de "Named Entity" (Ex. entity-type Personne)
- DURM (pour l'allemande)

Addendum : Les outils et la technologie existants (suite)

Gate :



Addendum : Les outils et la technologie existants (suite)

Quelques références :

- GATE :
  - http://gate.ac.uk
  - http://sourceforge.net/projects/gate
  - Docs : http://gate.ac.uk/sale/tao/
- UIMA :
  - http://www.research.ibm.com/UIMA/
  - http://sourceforge.net/projects/uima-framework/
- Autres :
  - WordNet: http://wordnet.princeton.edu/
  - MuNPEx: http://www.ipd.uka.de/d̃urm/tm/munpex/

Addendum : uUn exemple d'application médicale

Utilisation du TM pour découvrir des hypothèses médicales

- Dr Don Swanson s'est intéressé aux causes de la migraine.
- Il a compilé des articles sur le sujet dans la littérature biomédicale.
- Quelques une des pistes peuvent être présentées somme suit :
  - *Le Stress est associée à la migraine.*
  - *Le Stress peut conduire à une perte de magnésium .*
  - *Les bloqueurs de canaux Calciques (de Calcium, qui freinent l'entée du Calcium dans les cellules) préviennent certaines migraines.*
  - *Le Magnésium est un bloqueur de canaux Calciques naturel.*
  - *Le "Spreading Cortical Depression" (SCD = "dépression corticale envahissante" = le processus de propagation des ondes dans le cortex visuel) est impliquée dans certaines migraines. ➛ Les difficultés de "vision" des migraineux .*
  - *Un niveau élevé de Magnésium inhibe le SCD.*
  - *Les patients migraineux ont un niveau élevé "d'agrégation plaquettaire" (accumulation néfaste de plaquettes sur des cellules dans le cerveau)*
  - *Le magnésium peut supprimer ces agrégats de plaquettes.*

Addendum : uUn exemple d'application médicale (suite)

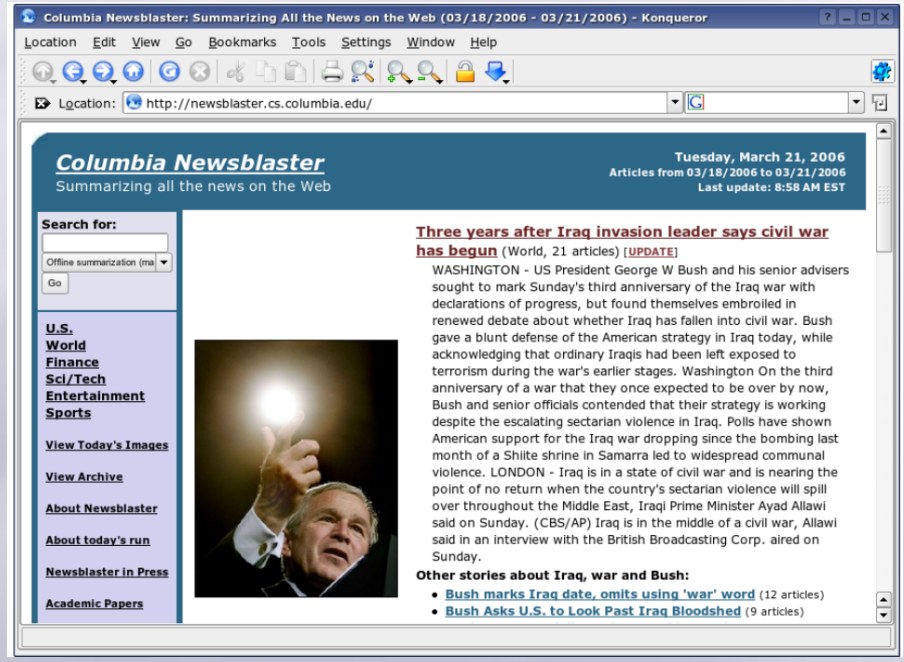
- Ces pistes suggèrent que la déficience en Magnésium peut jouer un rôle dans certaine migraines.
  - Cette hypothèse n'existait pas dans la littérature quand le Dr. Swanson l'a découverte.
  - Le point important est qu'une hypothèse nouvelle et médicalement plausible a été découverte à travers des fragments de texte par l'expertise de médecin.
  - Elle a été testé cliniquement
    [Swanson 1991, Ramadan 1989].

Addendum : Author Recognition & Verification

- Pour identifier quel auteur a écrit un document anonyme (non attribué)
- Les autres utilisations typiques comprennent :
  - Détection de plagiat d'auteur
  - Vérification d'une écriture revendiquée
  - Identification de courriels, des messages de *chat* ou d'un renseignement (secret) ....
- Plusieurs approches sont disponibles pour la vérification et la reconnaissance d'auteur
- Une technique nouvelle (à base du TM) **profilage linguistique** (Linguistic Profiling [John Baugh, 2003][van Halteran Hans, 2011]) a donné les performances :
  - 8.1 % de faux positif (FP) avec 0% faux négatif (FN) pour la vérification
  - Et 99,4 % de justesse de la reconnaissance (TP, dans les 2 sens : Auteur $\Longleftrightarrow$ doc)
- Technique basée sur l'accent / dialecte / ...

Addendum : Résumé automatique

- Trouver et résumer ce qui est important
  - Résumé à partir d'abstracts
  - Résume à partir de contenu
- **NewsBlaster** : site de résumé (pas un outil de résumé)



- **DUC summarization** : http://duc.nist.org

Addendum : Opinion Mining

- Ex : *amazon.com, Epinions.com*
- Difficile pour une entreprise de tracker les opinions $\rightarrow$ TM
- Axé sur le commerce et les produits
- Les Phases ([Popescu & Etzioni, HLT/EMNLP 2005]) :
  - Détecter les caractéristiques des produits p. ex. discutées dans les "review"s
  - Détecter les Opinions: en rapport avec ces caractéristiques
  - Déterminer la "Polarité" de ces opinions (positive / negative?)
  - Classer les opinions: basé sur leur force (comparer "so-so" vs. "desaster")
- Les outils NE (*Named Entity*) très utiles pour cette tâche.

Addendum : Q/A system

Système Q/R : une approche typique

➛ Les smartphones commencent à l'utiliser.

• La plupart des systèmes Q/A suivent à peu près un processus en trois étapes :

1 ◦ Étape de **Génération**: trouver des documents à partir d'un ensemble qui pourrait être pertinent pour la question

2 ◦ Etape de **Choix** de réponse : traiter les documents récupérés pour trouver des réponses possibles

3 ◦ Etape **formulation** de Réponse : créer une réponse dans le format requis (GN seul, phrase complète, etc.)

. . . ⇝

Addendum : Q/A system (suite)

- Pour trouver la réponse, une multitude d'approches existent :
  - **syntaxiques**: trouver des modèles correspondant ou analyser et produire un (sous-)arbre syntaxique (avec certaines transformations) à la fois de Q et de A
  - **sémantique**: transformer à la fois Q et A dans une forme logique et utiliser l'inférence (logique) pour vérifier la cohérence
  - Utiliser **Google**: poser la question dans Google et sélectionner la réponse avec une stratégie syntaxique ...

    $\ldots \rightsquigarrow$

- Voir aussi le BE pour l'utilisation de TF-Idf + SVD (pour les cas simples)

Addendum : Q/A system (suite)

Exemple Question Answering

- *Who invented the telephone?* ➛ Alexander Graham Bell
- *When was the telephone invented?* ➛ 1876

- Ex. : la démarche dans **QA System Shapaqa** :
  - Analyser la question :
    ➛ *When was the telephone invented?*
  - Les slots donnés :
    ➛ **Verb** *invented*
    ➛ **Object** *telephone*
  - Slot demandé ?
    ➛ *Temporal phrase linked to verb*
  - Google : on cherche sur Internet avec les slots donnés (mots clefs)
  - Analyse avec les slots donnés
  - Compter les entrées les plus fréquentes avec le slot demandé (temporal phrase) ... ⇝

Addendum : Q/A system (suite)

- *When was the telephone invented?*
  - Par Google : "invented" et "the telephone"
    - ➛ produit 835 pages
    - ➛ 53 phrases analysées avec les 2 slots et une phrase temporelle (fait par le système)

- **La réponse du système :**

  It is through his interest in Deafness and fascination with acoustics that the telephone was invented in 1876, with the intent of helping Deaf and hard of hearing.

  The telephone was invented by Alexander Graham Bell in 1876.

  When Alexander Graham Bell invented the telephone in 1876 , he hoped that these same electrical signals could ...

Addendum : Q/A system (suite)

Remarque : même chose mais sur Google simplement

- *So when was the phone invented ?*
  - La réponse obtenue : (via Internet, bruitée mais robuste)
    - ☞ 17: 1876
    - ☞ 3: 1874
    - ☞ 2: ago
    - ☞ 2: later
    - ☞ 1: Bell
    - ☞ ...

➛ Precision 76% (Google 31%)
➛ Compétition Internationale (TREC): MRR 0.45
➛ MRR : *Mean Reciprocal Rank Scoring*
➛ 0.45 est un score assez élevé

Addendum : Application en Biologie

- **BioRAT** : développé à University College London (UCL)
  - Outil de rech et extraction d'information en Biologie
  - Composé de :
    - ➛ Un outil IR (à la google)
    - ➛ Un outil d'extraction d'information basé sur GATE (Tokens, POS tags, ...)
    - ➛ Une template pour définir les motifs à extraire.
- **MutationMiner** (Protéine), utilise GATE
- ...

Addendum : Web Mining

☞ Une différence majeure de contexte : données (semi)-structurées.

- **Web content mining**
  Data Mining sur les données WEB
- **Web structure mining**
  Extraction d'infos utiles à partir des liens (links) dans les pages
- **Web usage mining**
  Extraction d'infos utiles par des données récupérées à partir des visites des pages, les transactions, ...
- **Utilité du Web mining** (en commerce) :
  - Déterminer la durée de vie des clients (au sens client !)
  - Conceptions de stratégies cross-marketing à travers les produits
  - Évaluation des campagnes de promotion
  - Ciblage des pubs électroniques,
    - ➛ Pour cibler les coupons / points vers les groupes d'utilisateurs
  - Prédire l'attitude / comportement des utilisateurs (clients)
  - Présenter des infos dynamiques aux clients

Addendum : Web Mining (suite)

Particularités du "Web Data Mining" ?

- La disponibilités de ces données et la valeur ajoutée qu'on peut en tirer.
- Web data mining = *screen scraping* = *web scraping* = *data extraction*
  extraction de connaissances à partir de sources semi-structurées (html/XML)

- Mais
  - Différent du texte, des données transactionnelles, les BDs
  - Le format html rend le texte lisible (pour l'homme).
  - Devient un obstacle (il faut enlever les balises) après avoir exploité la structure ➛ (entête, liste, ...)
  - La présence d'images et autres données multi-média posent problèmes pour une exploitation automatique.
  - **Il faut un pré-traitement**

- **Outils**
  - Essayer DOMZ,
  - Voir le site *Knugets* pour une liste d'outils libres.
  - Commerciaux : *SAS* , *Cognos*, voir aussi chez Microsoft.

- Pour une liste des logiciels (Commercial ou libre) :
  http://www.kdnuggets.com/software/suites.html

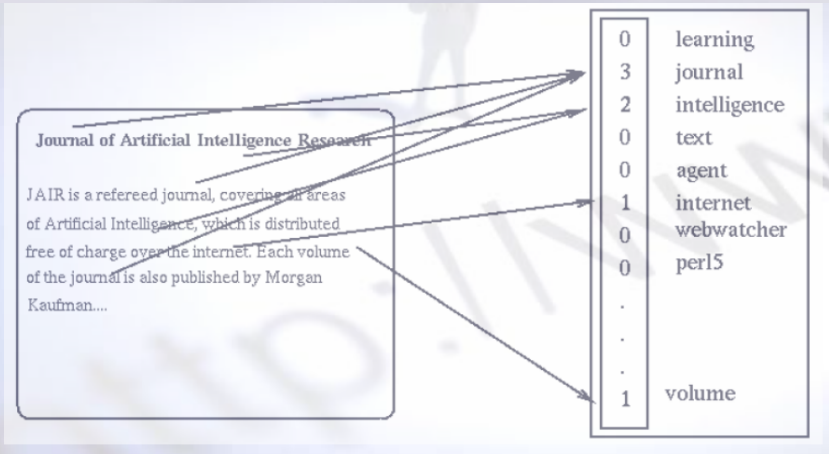
Quelques références

- Weiss, Indurkhya, Zhang, Damerau, Text Mining: Predictive Methods for Analyzing Unstructured Information, Springer, 2005.
- Sophia Ananiadou and John McNaught (Eds.), Text Mining for Biology and Biomedicine, Artech House, 2006. Inderjeet Mani, Automatic Summarization, John Benjamins B.V., 2001
- Cunningham, Information Extraction, Automatic, Encyclopedia of Language and Linguistics, 2005.
  http://gate.ac.uk/sale/ell2/ie/
- Zhong & Liu (Eds.), Intelligent Technologies for Information Analysis, Springer, 2004
- Peter Morville, Ambient Findability, O'Reilly, 2006
- Et beaucoup d'autres

Retour aux Techniques IR

- Aux côtés des techniques de "computational linguistics", on a des techniques plus simples.
- Deux approches :
  - Sac de termes / mots ("bag of words" = BOW)
  - Vecteur de caractéristiques (Vector space)

Sac de mots :



Grobelnik, M. and Mladenic, D. Text-Mining Tutorial. In the Proceeding of Learning Methods for Text Understanding and Mining, Grenoble, France, January 26 – 29, 2004.

Retour aux Techniques IR (suite)

Bag of words (BOW) :

- En général, on fait un sac des mots (des *uni-grammes*)
  - ➛ Pas de grammaire/POS/... ni l'ordre des mots (dommage !)
- On peut retenir la présence / la fréquence / le nombre d'occurrences de chaque mot
- On peut constituer un dictionnaire
- Une classification / clustering est possible sur la base des mots / termes et leurs occurrences.
- En général : **perte du contexte** (avec *uni-grammes*)
- Les outils de détection de spam (cf. *Bayes*) utilisent cette technique
  - ➛ Là où la présence / fréquence d'un mot compte plus que le sens
  - ➛ Là où on est intéressé par 2 distributions de mots : celle des "spam"s (illégitime) et celle des "ham"s (légitime)
- Il est possible de faire un sac de *bi-grammes* (3-grammes + rare).

• Dans la représentation "sac de mots", chaque mot est représenté comme une variable aléatoire avec une *importance*.

What this talk is about

- For Text (and Image)
  - IR
  - LSA/LSI
    - PCA & SVD
    - Key Phrase extraction
  - PLSA
  - LDA
  - ...

Traditional IR

- From textual analysis world (NLP) (called also IR in 80s)
- Trivially based on keyword matching + some work around ..
- Measure the **distance** between a **query** and docs
  - ➛ The query may be short, ambiguous, incomplete, with synonymous words, etc.
- Distance metric examples (x, y are doc. vectors) :
  - ➛ Euclidean distance $$d_e(x,y) = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2}$$
  - ➛ Cosine : $cos(\Theta_{xy}) = \frac{x.y}{|x||y|}$

    where $x.y$ is the *Dot-Product* $$x.y = \sum_{i=1}^{N} x_i y_i$$
  - ➛ Other distances...

Traditional IR (suite)

Traditional Problems :

- **keyword** Mismatch :
  matching based on **keywords** is not sufficient
- **Vocabulary** Mismatch : same concept may be described by different people with different vocabulary
- **Polysemy**: more than one distinct meaning
  ➛ decreases *precision* metric, see below.
- **Synonomy**: many ways to refer to the same object
  ➛ decreases *recall* metric.

- **Performances** evaluation : *Recall* and *Precision*

$$\text{Recall}=\frac{\#extracted\ relevants(\text{TP})}{\#total\ relevants(\text{TP}+\text{FN})}$$

$$\text{Precision}=\frac{\#extracted\ relevants(\text{TP})}{\#total\ extracted(\text{TP}+\text{FP})}$$

One solution : Vector space

One may use **Salton's Vector Space Model** to incorporates local and global information in similarity analysis.

- Count occurrences of a term in a doc : *#term* in a doc.
- Binary representation : $1 \text{ if } \text{term}_i \in \text{doc}_j; 0 \text{ otherwise.}$
- Term Weight : $\text{tfidf}(w, \text{d}) = \text{tf}(w, \text{d}).\left(\frac{|\text{D}|}{\text{df}(w)}\right)$
  - $\text{tf}(w, \text{d})$: term frequency (weight of w's occurrence in d = **local** information)
  - $\left(\frac{|\text{D}|}{\text{df}(w)}\right)$ called $\text{IDF}(w, \text{d})$: is the inverse document frequency
    ➛ a **global** information
  - $\text{df}(w)$: document frequency (#docs containing term w)
    ➛ $\text{df}(w) = |\text{d}' \in \text{D}, w \in \text{d}'|$ is a **global** information
  - $|\text{D}|$: *#docs* in the database.
- Other Term Weight : $\text{tfidf}(w, \text{d}) = 1 + \log(\text{tf}(w, \text{d})).\log\left(\frac{|\text{D}|}{\text{df}(w)}\right)$
- See also **Word2vect**

One solution : Vector space (suite)

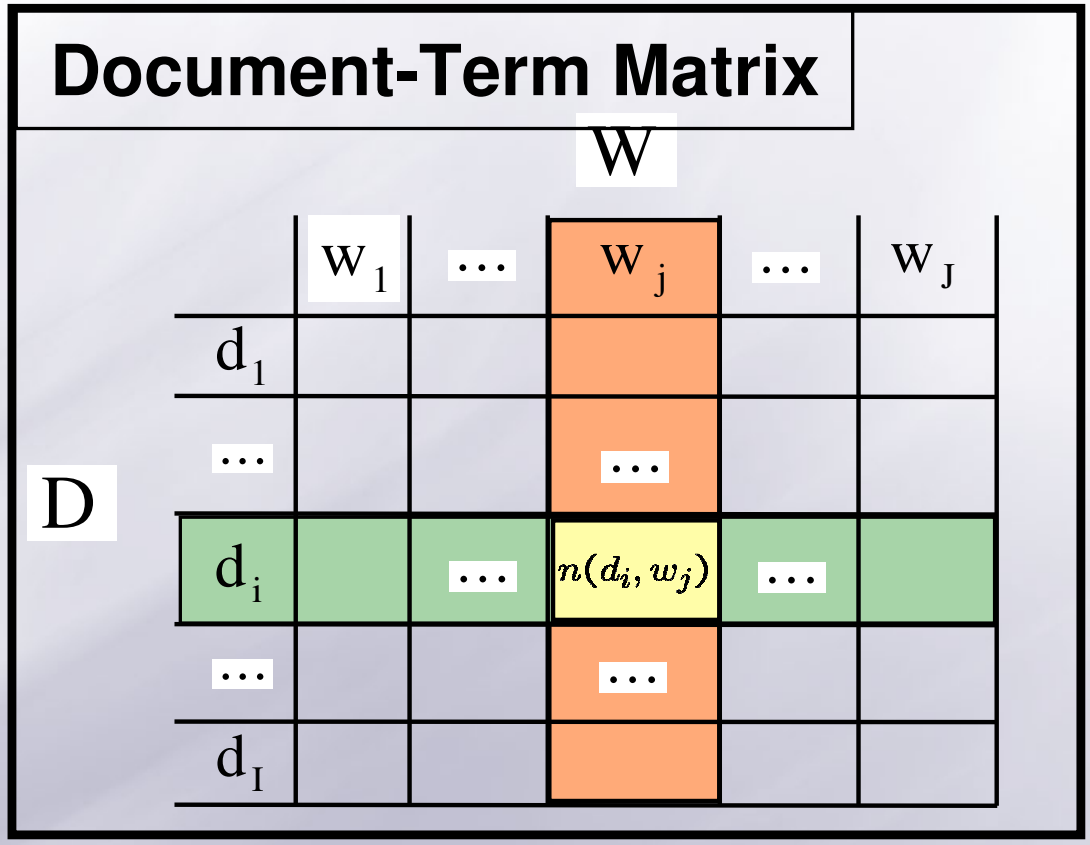


Figure 2: co-occurrence matrix

One solution : Vector space (suite)

- Exemple : le texte

TRUMP MAKES BID FOR CONTROL OF RESORTS Casino owner and real estate Donald Trump has offered to acquire all Class B common shares of Resorts International Inc, a spokesman for Trump said. The estate of late Resorts chairman James M. Crosby owns 340,783 of the 752,297 Class B shares. Resorts also has about 6,432,000 Class A common shares outstanding. Each Class B share has 100 times the voting power of a Class A share, giving the Class B stock about 93 pct of Resorts' voting power.

Le vecteur (les zéros non représentés)

[RESORTS:0.624] [CLASS:0.487] [TRUMP:0.367] [VOTING:0.171] [ESTATE:0.166] [POWER:0.134] [CROSBY:0.134] [CASINO:0.119] [DEVELOPER:0.118] [SHARES:0.117] [OWNER:0.102] [DONALD:0.097] [COMMON:0.093] [GIVING:0.081] [OWNS:0.080] [MAKES:0.078] [TIMES:0.075] [SHARE:0.072] [JAMES:0.070] [REAL:0.068] [CONTROL:0.065] [ACQUIRE:0.064] [OFFERED:0.063] [BID:0.063] [LATE:0.062] [OUTSTANDING:0.056] [SPOKESMAN:0.049] [CHAIRMAN:0.049] [INTERNATIONAL:0.041] [STOCK:0.035] [YORK:0.035] [PCT:0.022] [MARCH:0.011]

One solution : Vector space (suite)

- Example (from Dr. Grossman) of **TfIdf**

TERM VECTOR MODEL BASED ON $w_i = tf_i * IDF_i$

Query, Q: "gold silver truck"
D_1: "Shipment of gold damaged in a fire"
D_2: "Delivery of silver arrived in a silver truck"
D_3: "Shipment of gold arrived in a truck"
$D = 3$; $IDF = \log(D/df_i)$

		Counts, tf_i						Weights, $w_i = tf_i * IDF_i$			
Terms	Q	D_1	D_2	D_3	df_i	D/df_i	IDF_i	Q	D_1	D_2	D_3
a	0	1	1	1	3	3/3 = 1	0	0	0	0	0
arrived	0	0	1	1	2	3/2 = 1.5	0.1761	0	0	0.1761	0.1761
damaged	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
delivery	0	0	1	0	1	3/1 = 3	0.4771	0	0	0.4771	0
fire	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
gold	1	1	0	1	2	3/2 = 1.5	0.1761	0.1761	0.1761	0	0.1761
in	0	1	1	1	3	3/3 = 1	0	0	0	0	0
of	0	1	1	1	3	3/3 = 1	0	0	0	0	0
silver	1	0	2	0	1	3/1 = 3	0.4771	0.4771	0	0.9542	0
shipment	0	1	0	1	2	3/2 = 1.5	0.1761	0	0.1761	0	0.1761
truck	1	0	1	1	2	3/2 = 1.5	0.1761	0.1761	0	0.1761	0.1761

One solution : Vector space (suite)

Similarity analysis (between a query and a document)

➛ Cosine similarity distance :

$$Sim(Q, D_i) = Cosine\,\Theta_{D_i} = \frac{Q.D_i}{|Q||D_i|} = \frac{\sum_j w_{Q,j} w_{i,j}}{\sqrt{\sum_j w_{Q,j}^2}\sqrt{\sum_j w_{i,j}^2}}$$

- For the above example, one obtains the ranking (vs. query Q):
  Rank 1: Doc 2 = 0.8246 ← Cosine(Q, D2) is max.
  Rank 2: Doc 3 = 0.3271
  Rank 3: Doc 1 = 0.0801

  ➛ Q is more similar to Doc 2 (based on TfIdf)

- `TfIdf` in a vector space method is simple but weak.
- Towards LSA . . .

LSA / LSI and its computational tools

- **Try to overcome problems of IR** by doing LSA Based on co-occurrence matrix.
- LSA assumes that there exists a LATENT structure in word usage, obscured by the variability in a word choice

- **Tools**
  - **PCA** : get the principal components by reducing data points dimension (while preserving information)
  - ➔ Choose projections that minimize the squared error (distance) of the projection vs. the original data
  - ➔ I.e., Get eigenvectors corresponding to the largest eigenvalues of the covariance matrix : preserve the highest variance of the Data.
  - **SVD** : see the example...

LSA / LSI and its computational tools (suite)

- **General idea of LSA**
  - Map documents (and terms) to a lower-dimensional representation.
  - Design a mapping such that the lower-dimensional space reflects semantic associations (*latent semantic space*).
  - Compute document similarity based on the inner product in the latent semantic space

- **Goals**
  - "Similar" terms (similarity measure) map to similar location in the lower dimensional space
  - Get noise-reduction by dimension reduction

From PCA to SVD

• PCA tries to fit an n-dimensional ellipsoid to the data, where each axis of the ellipsoid represents a **principal component**.

☞ If some axis of the ellipse is small, then the variance along that axis is also small, and by omitting that axis and its corresponding principal component from our dataset, we lose only a small amount of info.

How PCA works :

• For $x_1, ..., x_n \in \mathbb{R}^p$ a set of *centered* (with $\mu_{w_i} = 0$) points, find a k-dimensional subspace with the least loss of information.

• PCA finds a $n \times k$ (rather than $n \times p$) linear projection matrix V_k so that the sum of squared dist. from the data pts. to their proj. is minimized :

$$\text{Loss cost} : \mathfrak{L}(V_k) = \sum_{i=1}^{n} \|x_i - V_k V_k^T x_i\|^2 \qquad x_i \text{ is a doc. for us}$$

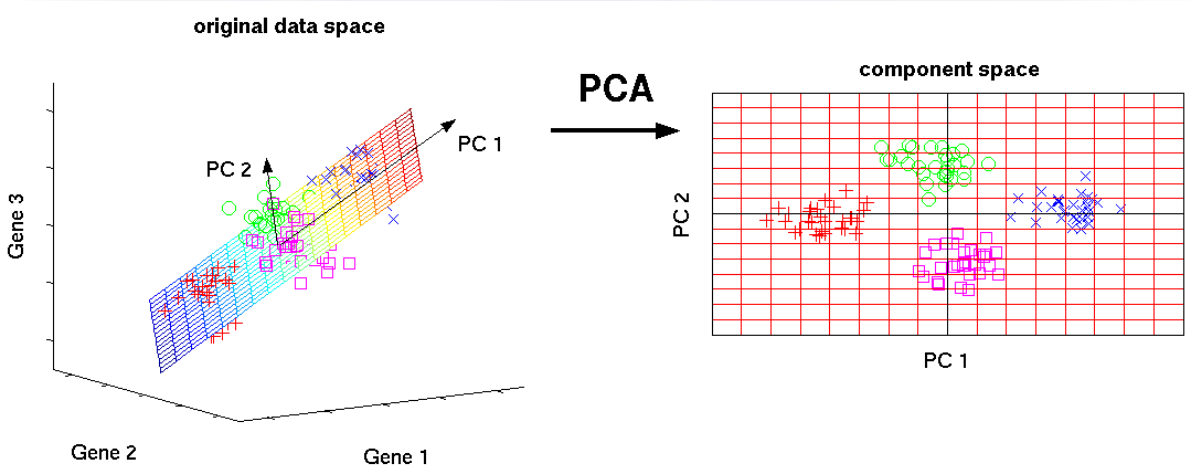
$V_k^T x_i$: projection of x_i in the k-subspace spanned by the column vectors of V_k

$V_k V_k^T x_i$: the (new) repr. in the original p-space of the projected vector $V_k^T x_i$

☞ PCA supports only sum-square distance, not *Cosine*, etc. (convergence issue)

From PCA to SVD (suite)

Example : PCA and Bioinformatics



- A three-dimensional gene expression data mainly located within a two-dimensional subspace.
- The **three original variables** (genes) are reduced to a lower number of **two new variables** termed principal components (PCs).
- Using PCA, we can identify the two-dimensional plane that optimally describes the highest variance of the data.
- This two-dimensional subspace can then be rotated and presented as a two-dimensional component space (right).
- This will allow us to draw qualitative conclusions about the **separability** of experimental conditions (marked by different colors).

From PCA to SVD (suite)

In PCA (Cont.) :

- We have to minimize the cost $\mathfrak{L}(V_k)$:

$$\mathfrak{L}(V_k) = \sum_{i=1}^{n} \|x_i - V_k V_k^T x_i\|^2 = \sum_{i=1}^{n} \|x_i\|^2 - \sum_{i=1}^{n} \|V_k V_k^T x_i\|^2 \qquad \text{where}$$

$\sum_{i=1}^{n} \| V_k V_k^T x_i \|^2$: the empirical variance of the projections.

and $V_k^T V_k = Id \quad \Leftrightarrow \quad V_k$ is orthogonal

☞ The projection matrix V_k that minimizes $\mathfrak{L}(V_k)$ is the one that maximizes the variance in the projected space.

- **The solution to V_k can be computed by SVD.**
- We can also use SVD directly.

Addendum : PCA details, PLSI

why $\mathfrak{L}(V_k) = \sum_{i=1}^{n} ||x_i - V_k V_k^T x_i||^2 = \sum_{i=1}^{n} ||x_i||^2 - \sum_{i=1}^{n} ||V_k V_k^T x_i||^2$?

➛ $||x_i - V_k V_k^T x_i||^2 =< x_i - V_k V_k^T x_i \ , \ x_i - V_k V_k^T x_i >$

$= ||x_i||^2 - 2 < x_i, V_k V_k^T x_i > + ||V_k V_k^T x_i||^2$

$= ||x_i||^2 - 2 < V_k^T x_i, V_k^T x_i > + ||V_k V_k^T x_i||^2$ since $< A, B >=< CA, CB >$ if $C^T C = Id$

$= ||x_i||^2 - 2||V_k^T x_i||^2 + < V_k V_k^T x_i, V_k V_k^T x_i >$ And $V_k^T V_k = Id$

$= ||x_i||^2 - 2||V_k^T x_i||^2 + < V_k^T V_k V_k^T x_i, V_k^T x_i >$ same as above

$= ||x_i||^2 - 2||V_k^T x_i||^2 + < V_k^T x_i, V_k^T x_i >$ ☞ $V_k^T V_k = Id$

$= ||x_i||^2 - 2||V_k^T x_i||^2 + ||V_k^T x_i||^2$ ☞ But $V_k V_k^T \neq Id$ otherwise there is no loose !

$= ||x_i||^2 - ||V_k^T x_i||^2 = ||x_i||^2 - ||V_k V_k^T x_i||^2$ We had $||V_k V_k^T x_i||^2 = ||V_k^T x_i||^2$ (from lines 3 to 7)

- This shows that $||V_k V_k^T x_i|| = ||V_k^T x_i||$
  - ➛ But we better use $||V_k V_k^T x_i||$ to handle the cost function.

- Note that for any matrix $X, XX^T = ||X||^2$ is symetric

Addendum : PCA details, PLSI (suite)

More details about PCA :

• To find the axes of the ellipse in **PCA**, first **center** the data around the origin (i.e. subtract the mean of each variable from the dataset).

Then compute the covariance matrix of the data, and calculate the eigenvalues and corresponding eigenvectors of this covariance matrix.

Then, orthogonalize the set of eigenvectors, and normalize each to become unit vectors.

Once this is done, each of the mutually orthogonal, unit eigenvectors can be interpreted as an axis of the ellipsoid fitted to the data.

The proportion of the variance that each eigenvector represents can be calculated by dividing the eigenvalue corresponding to that eigenvector by the sum of all eigenvalues.

☞ Note that PCA is sensitive to the scaling of the data : this is a limitation of PCA
➛ there is no consensus as to how to best scale the data to obtain optimal results.

• **Mathematically**, PCA is defined as an orthogonal linear transformation of the data to a new coordinate system such that the greatest **variance** by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

➛ The principal components are orthogonal because they are the eigenvectors of the covariance matrix, which is symmetric.

• PCA is computed by using the **correlation** or the **covariance** method

Addendum : PCA details, PLSI (suite)

Towards PLS (and NIPALS) :
PCA improvement : the NIPALS method (Non-linear iterative partial least squares)

- For very-high-dimensional datasets (e.g., genomics, metabolomics) it is usually only necessary to compute the first few PCs.

  The non-linear iterative partial least squares (NIPALS) algorithm calculates the first components from the data set.
  Then this component is "subtracted" from the data set leaving a residual matrix.
  This can be then used to calculate subsequent PCs.
  This results in a real reduction in computational time since calculation of the covariance matrix is avoided.

  ➛ However, **for large data matrices**, or matrices that have a high degree of column collinearity, NIPALS suffers from loss of orthogonality due to machine precision limitations accumulated in each iteration step.

  ➛ A Gram-Schmidt (GS) re-orthogonalization algorithm is applied to both the scores and the loadings at each iteration step to eliminate this loss of orthogonality.

- For an online/sequential (streaming) estimation, a different method must be applied.

SVD (singular value decomposition)

The aim of SVD is to find a diagonal matrix, to reduce dimensionality of the original co-occurrence matrix (np) to (nk) and use it instead of the original matrix (the data set).

- SVD : given the matrix A with $x_1, ..., x_n \in \mathbb{R}^p$

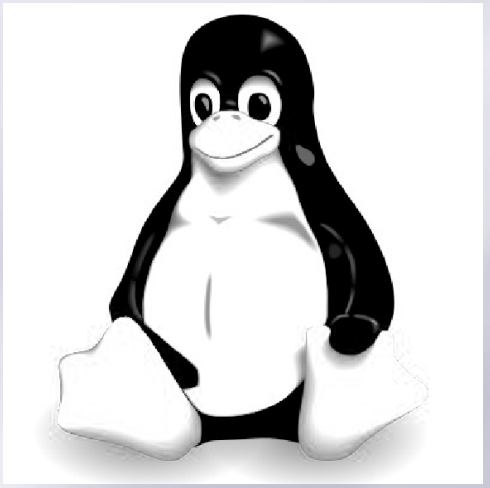
Let the SVD of matrix A be given by $A = USV^T$ with
$U = [u_1, u_2, ..., u_p]$,
$S = Diag[s_1, s_2, ..., s_p]$,
$V = [v_1, v_2, ..., v_p]$
and $rank(A) = r$.

Matrix A_k defined by $A_k = \sum_{i=1}^{k} u_i s_i v_i^T$ is the closest rank-k matrix to A in terme of Euclidien (Frobenious) norm.

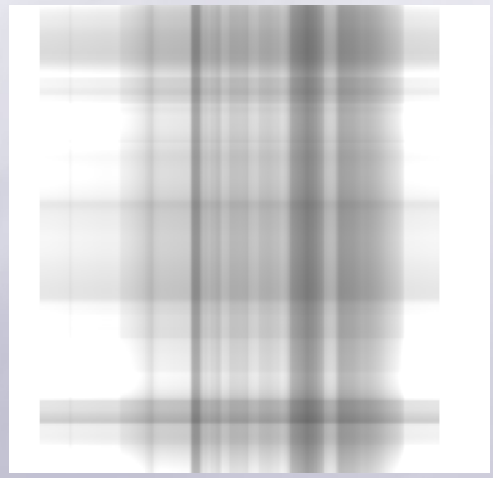
- For k given, the matrix V_k (in k-subspace and composed of the first k columns of V) constitutes **the rank k solution to** $\mathfrak{L}(V_k)$.

SVD (singular value decomposition) (suite)

- SVD with Tux
  (see https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Dimensionality_Reduction/Singular_Value_Decomposition)



Original



With one SV

with 35 SV

SVD (singular value decomposition) (suite)

- Main contributions of SVD ($A = USV^T$) :
  - Reducing dimension of A : the k-largest singular values (diagonal of S) approximates the original matrix A.
  - Discarding small singular values $\equiv$ discarding semi-dependent, noisy, trivial variation in the data set ($\equiv$ discard non-essential axes of the original feature space)
    - Emphesizes the salient underlying structure of the data set
    - Data points with similar features will be mapped near to each other in the k-dimensional partial singular vector space.

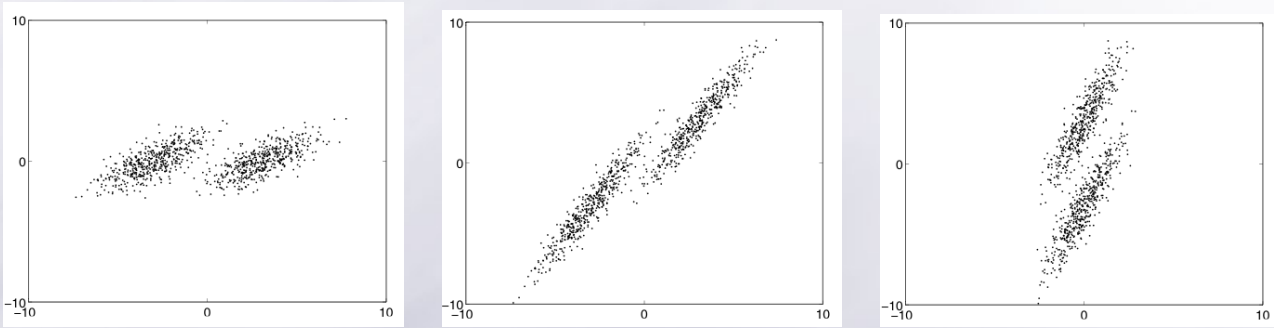
☞ In $A = USV^T$ (A is the co-occ term-doc matrix) :
- The matrix U_k represent the projection of Terms in the k-space
- The matrix V_k represent the projection of Docs in the k-space

➛ See further farther (the example)

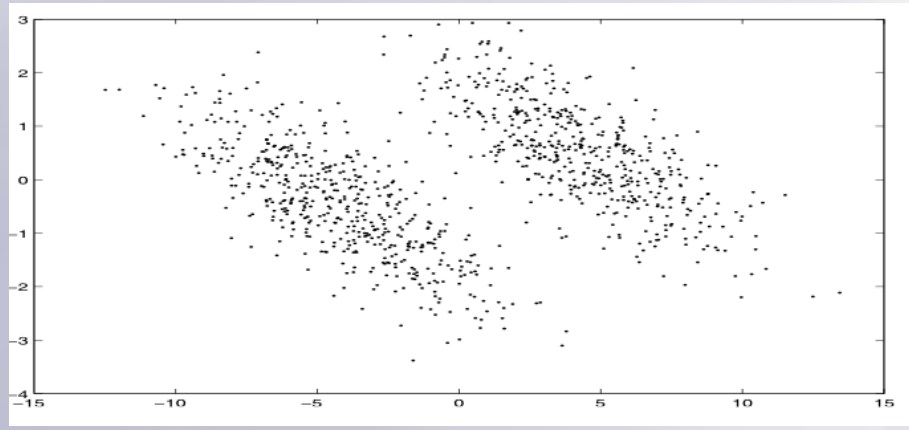
SVD (singular value decomposition) (suite)

An example of dimensionality reduction:

3 sub spaces X-Y, X-Z, Y-Z of a normal distribution 3D space



And the sub space spanned by the first 2 principal components (by SVD, similar to PCA) :



SVD (singular value decomposition) (suite)

- Un exemple on Image (thanks to Matworks.com) :

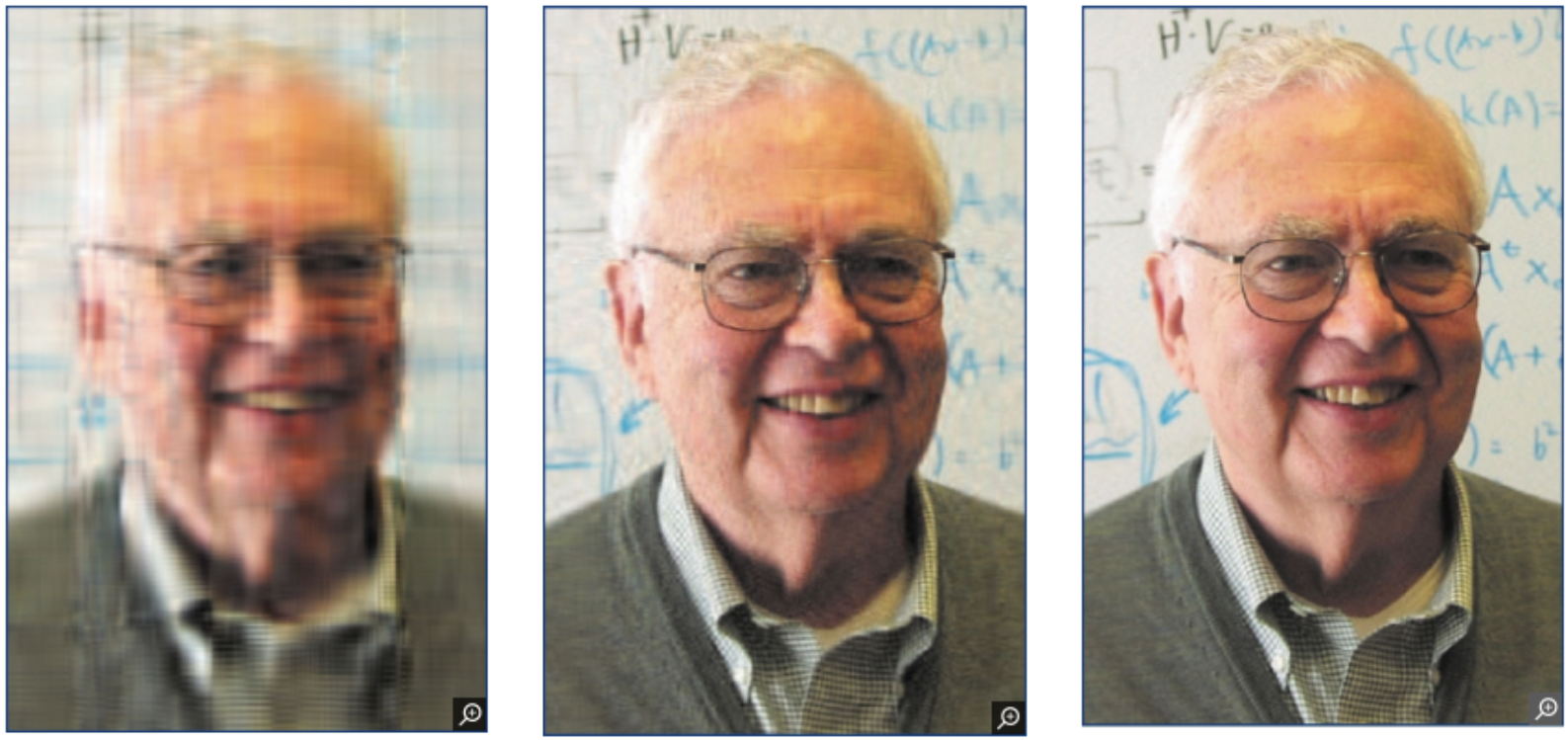


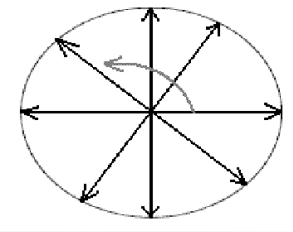
Figure 3: Rank 12, 50, and 120 approximations to a rank 598 color photo of Gene Golub.

LSI

- By using SVD for dimensionality reduction, we use LSA (and LSI)
  ➛LSA : latent semantics Analysis, LSI : latent semantics Indexing
- LSA is greate at addressing **synonymy** (different words with same meaning) but can fail to addresse **polysemy** (same word with different meanings).
- Also, the dimensionality reduction can incorrectly conflate critical dimensions.
- Selection of k values and the right clusters (performed later) is done arbitrarily, by-trial-and-error,
- Several workarounds have been proposed for LSA/LSI cons.
- LSA addresses *synonyms* but not *polysems* (despite a popular opinion)
  ➛ **A solution** : VLSI (variable latent Semantic Indexing).

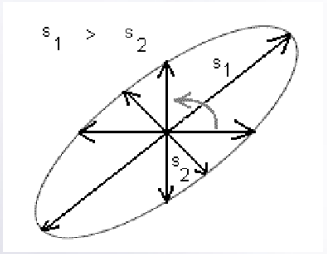
Addendum : understanding SVD

The geometrical interpretation of SVD :



- A simple analogy.
- Depict an arrow rotating in two dimensions and describing a circle.

➛ A rotating arrow of fixed length describing a circle :



- Now suppose that when rotating this arrow we stretch and squeeze it in a variable manner and up to a maximum and minimum length.

➛ Instead of a circle the arrow now describes a two-dimensional **ellipsoid**.

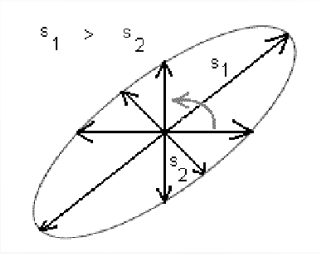
➛ This is exactly what a matrix does to a vector.

- When we multiply a vector X by a matrix A to create a new vector $AX = Y$, the matrix performs two operations on the vector:
  - rotation (the vector changes coordinates) and
  - scaling (the length of the vector changes).

- In terms of SVD, the maximum stretching and minimum squeezing is determined by the **singular values** of the matrix.

  ➛ These are values inherent, unique or "singular" to A that can be uncovered by the SVD algorithm.

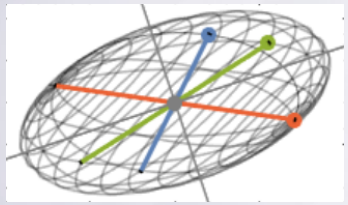
Addendum : understanding SVD (suite)



- In the figure the effect of the two largest singular values, $s1$ and $s2$ has been labeled.
- Singular values describe the extent by which the matrix distorts the original vector and, thus, can be used to highlight which dimension(s) is/are affected the most by the matrix.
- Evidently, the largest singular value has the greatest influence on the overall dimensionality of the ellipsoid.
- In geometric terms, figures above show that multipliying a vector by a matrix is **equivalent** to transforming (mapping) a circle into an ellipsoid.
- Now, instead of a rotating, if the vector depicts a set of m-dimensional vectors, all perpendiculars (orthogonal), with different lengths and defining a m-dimensional ellipsoid :

➛ The ellipsoid is embedded in an m-dimensional space, where $\mathrm{m} = \mathrm{k}$, with k being the number of nonzero singular values.

- Note that a rotating arrow of variable length in a 3D space describes a 3D ellipsoid :



- If we eliminate dimensions by keeping the three largest singular values, a three-dimensional ellipsoid is obtained.

➛ This is a Rank 3 Approximation.

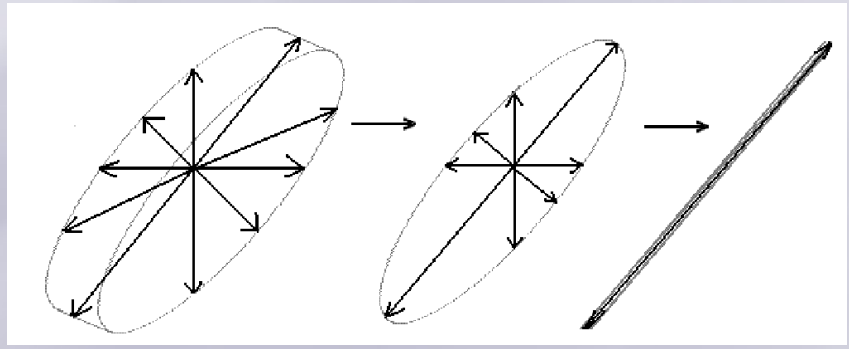
Addendum : understanding SVD (suite)

Dimensionality Reduction (by SVD) :

- Now suppose that compared with the two largest singular values the third one is small enough that we ignore it.
  - ➛ This is equivalent to removing one dimension.

- Geometrically, the 3-dimensional ellipsoid collapses into a 2-dimensional ellipsoid, and we obtain a Rank 2 Approximation.

- And, if we keep only one singular value the ellipsoid collapses into a one-dimensional shape (a line).
  - ➛ The following Figure illustrates roughly the reduction.



- This is why we say that keeping the top k singular values is a **dimensionality reduction** process.

Addendum : understanding SVD (suite)

- The main idea is this: singular values can be used to highlight which dimensions are affected the most when a vector is multiplied by a matrix.
  - ➛ Thus, the decomposition allows us to determine which singular values can be retained, from here the expression "singular value decomposition".

- In the case of LSI, SVD unveils the hidden or "latent" data structure, where terms, documents, and queries are embedded in the same low-dimensional and critical space.

This is a significant departure from term vector models.

- **In the vector space model** each unique term from an index term is considered a dimension of a term space.

- Documents and queries are then represented as vectors in this term space.

- In this space terms are considered independent from each other.

- However, this is contrary with common knowledge since term dependency can arise in written or spoken language in many different ways.
  - ➛ The most common are: through (a) synonymity and (b) polysemy.

Addendum : understanding SVD (suite)

• LSI performs better than vector space (Tfldf) models by embedding documents, terms and queries in the same space.

➛ This resolves the problem of high-order co-occurrence like *in-transit co-occurrence*.

In-transit co-occurrence accounts for the fact that terms do not need to be actually in a document to be relevant to its **content**. They can co-occur while "in transit".

An example of this are synonyms.

If we dig some more :

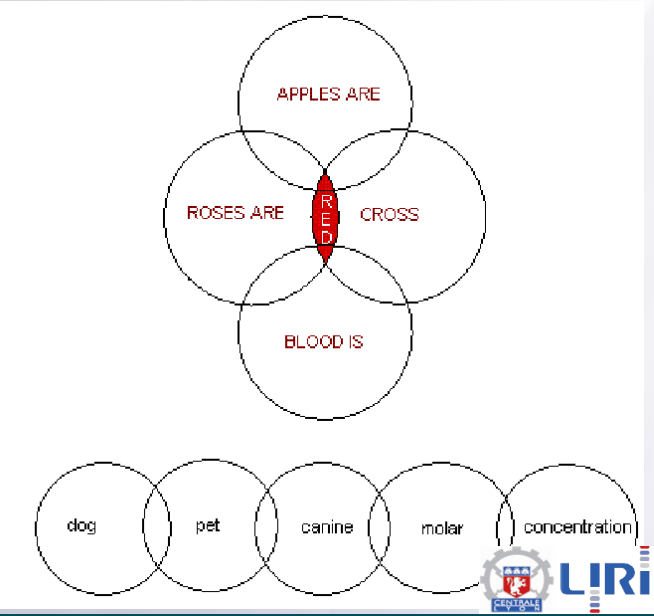
• Synonyms don't tend to co-occur together (often they have both low pairwise c-indices and low *Jaccard Indices* from a cluster association matrix) but they tend to co-occur in the same context : this can be inspected by measuring their high-order c-indices.

◦ Terms not related by a synonymity association can be in-transit and their high order co-occurrence can be examined with c-indices (illustrated in the Figure →).

◦ The *Venn* Diagrams (depicted in the figure) represent documents relevant to specific terms.

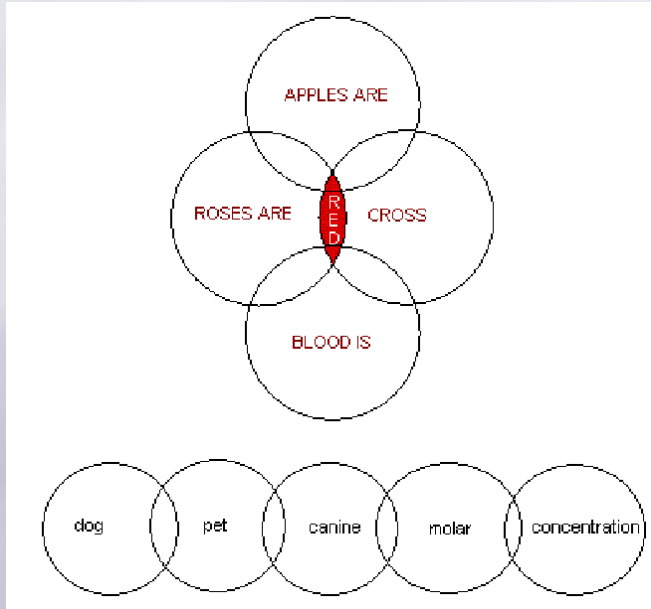
◦ The **(dog, canine)** and **(canine, molar)** pairs have a synonymity association, but this association is not present in the **(dog, molar)** pair or between the other pairs.

◦ Note that "In-transit" co-occurrence is not unique to synonyms.



Addendum : understanding SVD (suite)

- The figure shows also that in-transit co-occurrence not only can be observed between individual tokens, but between distinct contexts.
- Figure : In-transit co-occurrence between tokens and contexts.



To simplify, in the figure we have ignored other forms of co-occurrences between n-grams like multiple overlapping regions, which can actually give rise to an LSI "curse".

LSI : a simple example (SVD application)

From LSA (general) to LSI : Latent Semantic Indexing (LSI) in the context of IR (LSI Patented in 1989 : http://lsi.argreenhouse.com/lsi)

Purpose : use LSI to cluster terms.

➛ Find also terms that could be used to expand or reformulate the query.

The textual corpus (we had) $\{D_1, D_2, D_3\}$:

D_1: "Shipment of gold damaged in a fire"

D_2: "Delivery of silver arrived in a silver truck"

D_3: "Shipment of gold arrived in a truck"

The query : **gold silver truck**.

LSI : a simple example (SVD application) (suite)

Step 1: Score term weights (#occ.) and construct the term-document matrix A and the query matrix:

(Here #occ. but *Term Weight* : $w_i = \mathrm{tf}_i \times \log(\frac{\mathrm{D}}{\mathrm{df}_i})$ can be used)

Terms		d1	d2	d3		q
a		1	1	1		0
arrived		0	1	1		0
damaged		1	0	0		0
delivery		0	1	0		0
fire		1	0	0		0
gold	**A =**	1	0	1	**q =**	1
in		1	1	1		0
of		1	1	1		0
shipment		1	0	1		0
silver		0	2	0		1
truck		0	1	1		1

Figure 4: Original data co-occurrence matrix A and the query q

LSI : a simple example (SVD application) (suite)

Step 2: Decompose matrix A such that $A = USV^T$

U

-0.4201	-0.075	-0.046
-0.2995	0.2001	0.4078
-0.1206	-0.275	-0.4538
-0.1576	0.3046	-0.2006
-0.1206	-0.275	-0.4538
-0.2626	-0.379	0.1547
-0.4201	-0.075	-0.046
-0.4201	-0.075	-0.046
-0.2626	-0.379	0.1547
-0.3151	0.6093	-0.4013
-0.2995	0.2001	0.4078

S

4.0989	0	0
0	2.3616	0
0	0	1.2737

V^T

-0.4945	-0.6458	-0.5817
-0.6492	0.7194	-0.2469
-0.578	-0.2556	0.775

LSI : a simple example (SVD application) (suite)

Remember : In $A = U S V^T$

- Rows of V hold **eigenvector values**.
  These are coordinates of individual term vectors in Docs.
  - ➛ We may compare documents (Cosine similarity) line by line.
  - ➛ V (and V_k) is the document-TOPIC similarity matrice.
- U is Term-vector coordinates :
  - ➛ we may compare 2 terms by Cosine (is not Synonymy)
  - ➛ See e.g. lines 6 and 9 : *gold* and *shipment* similarity.
  - ➛ U (and U_k) is a term-TOPIC similarity matrice

$$U = \begin{bmatrix} -0.4201 & -0.075 & -0.046 \\ -0.2995 & 0.2001 & 0.4078 \\ -0.1206 & -0.275 & -0.4538 \\ -0.1576 & 0.3046 & -0.2006 \\ -0.1206 & -0.275 & -0.4538 \\ -0.2626 & -0.379 & 0.1547 \\ -0.4201 & -0.075 & -0.046 \\ -0.4201 & -0.075 & -0.046 \\ -0.2626 & -0.379 & 0.1547 \\ -0.3151 & 0.6093 & -0.4013 \\ -0.2995 & 0.2001 & 0.4078 \end{bmatrix}$$

$$S = \begin{bmatrix} 4.0989 & 0 & 0 \\ 0 & 2.3616 & 0 \\ 0 & 0 & 1.2737 \end{bmatrix}$$

$$V^T = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ -0.6492 & 0.7194 & -0.2469 \\ -0.578 & -0.2556 & 0.775 \end{bmatrix}$$

LSI : a simple example (SVD application) (suite)

Step 3: Reduce to a rank $k = 2$ approximation by keeping the first 2 columns of U and V and the first 2 columns and rows of S.

$$U \approx U_k = \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \quad k = 2$$

$$S \approx S_k = \begin{bmatrix} 4.0989 & 0.0000 \\ 0.0000 & 2.3616 \end{bmatrix}$$

$$V \approx V_k = \begin{bmatrix} -0.4945 & 0.6492 \\ -0.6458 & -0.7194 \\ -0.5817 & 0.2469 \end{bmatrix} \quad V^T \approx V_k^T = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \end{bmatrix}$$

Figure 5: Dim reduction (K = 2 instead of 3)

LSI : a simple example (SVD application) (suite)

Step 4: Find the new term vector coordinates in this reduced 2-dimensional space.

Rows of V : eigenvector values (coordinates of individual term vectors).

U_k : reduced Term-vector coordinates is recalled at right ↘

➛ E.g. take a look at the similarity of lines **gold** and **shipment**

$$S = \begin{bmatrix} 4.0989 & 0 & 0 \\ 0 & 2.3616 & 0 \\ 0 & 0 & 1.2737 \end{bmatrix} \qquad V^T = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ -0.6492 & 0.7194 & -0.2469 \\ -0.578 & -0.2556 & 0.775 \end{bmatrix}$$

Terms	Term Vector Coordinates	
a	-0.4201	0.0748
arrived	-0.2995	-0.2001
damaged	-0.1206	0.2749
delivery	-0.1576	-0.3046
fire	-0.1206	0.2749
gold	-0.2626	0.3794
in	-0.4201	0.0748
of	-0.4201	0.0748
shipment	-0.2626	0.3794
silver	-0.3151	-0.6093
truck	-0.2995	-0.2001

Thus (☞ $d'_i = d_i.U_k.S_k^{-1}$) :

- $d'_1(-0.4945, -0.6492)$
- $d'_2(-0.6458, 0.7194)$
- $d'_3(-0.5817, -0.2469)$

LSI : a simple example (SVD application) (suite)

In LSI, the query is treated as another document ($d = d^T U S^{-1}$).

Step 5: Find the new **query** vector coordinates in the reduced k-dimensional sub space (here, $k = 2$) using $q_k = q^T U_k S_k^{-1}$

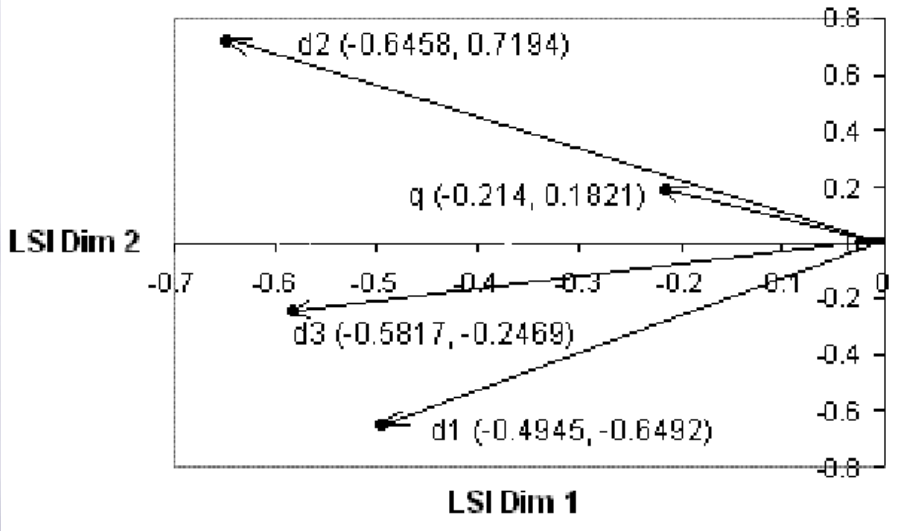
- ➛ These are new coordinates of the query in 2-dim space.
- ➛ Note the difference between the new one (at right) and the original q of step 1 (at left).

$$q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \begin{bmatrix} \frac{1}{4.0989} & 0.0000 \\ 0.0000 & \frac{1}{2.3616} \end{bmatrix} = \begin{bmatrix} -0.2140 & -0.1821 \end{bmatrix}$$

☞ We can also use $q_k = U_k^T . q$ (and $d_i' = U_k^T . d_i$).

LSI : a simple example (SVD application) (suite)

Step 6: Cosine similarity (OK) :



The textual corpus $\{d_1, d_2, d_3\}$:

d_1: "Shipment of gold damaged in a fire"
d_2: "Delivery of silver arrived in a silver truck"
d_3: "Shipment of gold arrived in a truck"

The query : **"gold silver truck"**

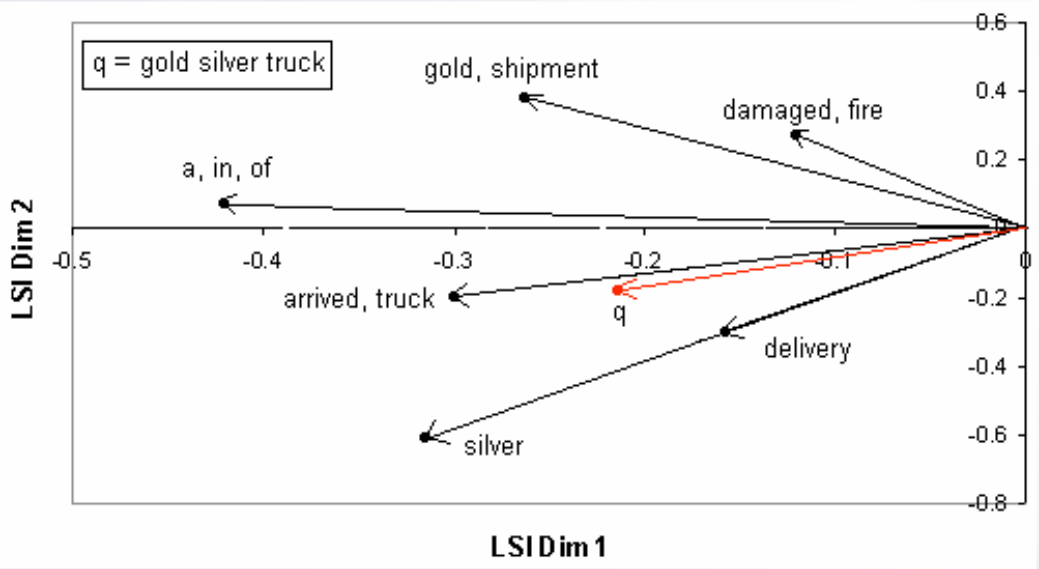
$$\mathbf{q} = \begin{bmatrix} -0.2140 & 0.1821 \end{bmatrix}$$

rather near to d_2

Term clusters & Query Expansion

Step I. Group terms into clusters (giving Vector coordinates).

- Clustring by comparing angles' Cosine between any two pair of vectors of U_k
- We can use (e.g.) **K-means**.



- Remember (Doc similarity) : we had $d_2 > d_3 > d_1$.

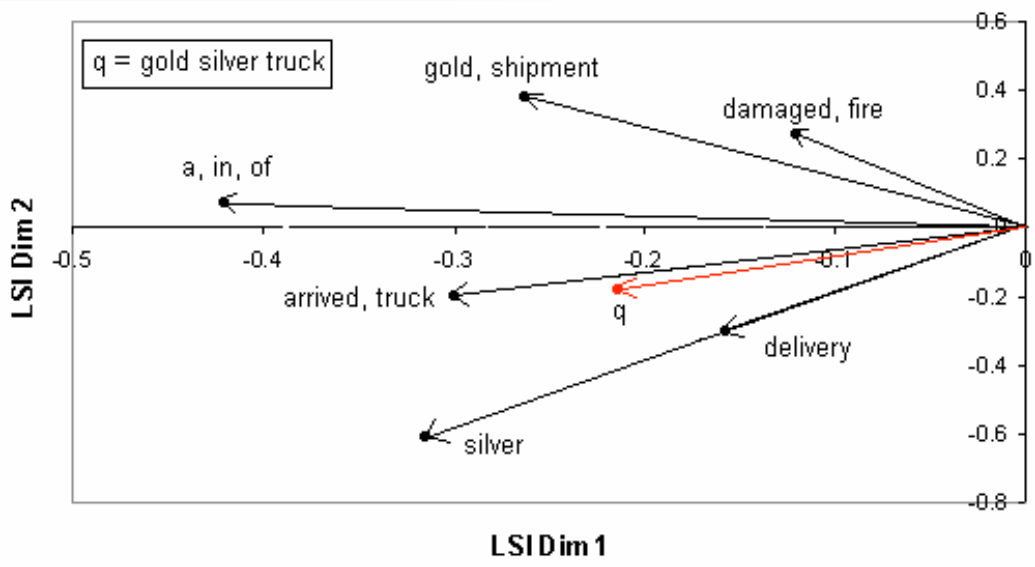
  ➛ Cosine similarity : $Cosine\ \Theta_{d_i} = \frac{q.d_i}{|q||d_i|}$

  For K-means, use $sim(q, d) = sim(q^T\ U_k\ S_k^{-1}, d^T\ U_k\ S_k^{-1})$

Term clusters & Query Expansion (suite)

Step I (cont.) : The following clusters are obtained:

1. *a, in, of*
2. *gold, shipment*
3. *damaged, fire*
4. *arrived, truck*
5. *silver*
6. *delivery*



NB. : Some vectors are not shown since these are completely superimposed / overlapped (those with > 1 term).

➛ If unit vectors are used (narmalization) and small deviation ignored, clusters 3 & 4 and clusters 4 & 5 can be merged.

Term clusters & Query Expansion (suite)

Step II. Find terms that could be used to expand or reformulate the query.

Related to the query "gold silver truck" :

- ➛ Clusters 1, 2 and 3 are far away from the query.
- ➛ Clusters are 4, 5, and 6 are closer.
- ➛ We could use 4, 5, 6 to expand or reformulate the query.

Some examples of the expanded queries that one can test :

"gold silver truck arrived " *"delivery gold silver truck"*

"gold silver truck delivery "

"gold silver truck delivery arrived " etc ...

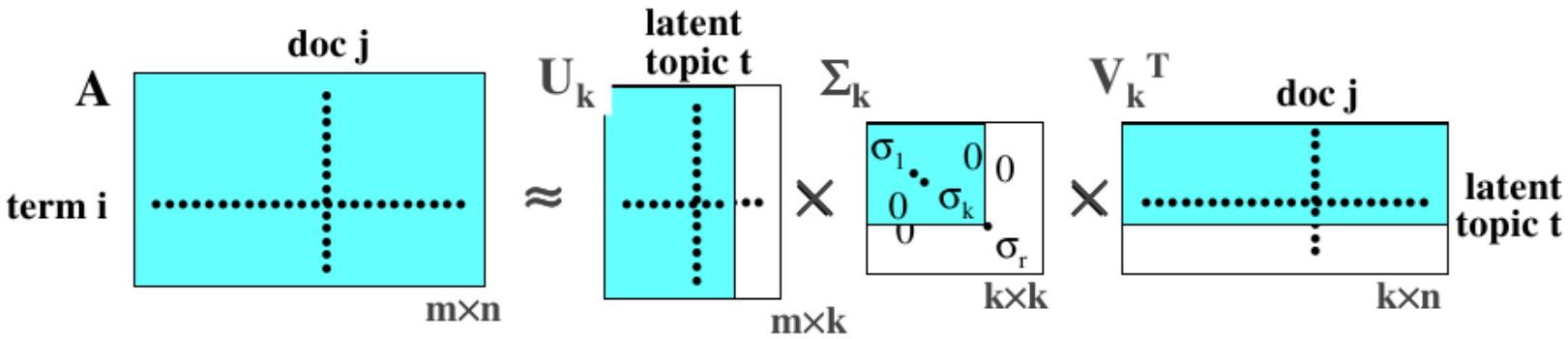
Documents containing these terms should be more relevant to the initial query.

Ask me something, I'll tell you what you mean !

LSA in some words

A is the m×n term-document similarity matrix. Then:

- U and U_k are the m×r and m×k term-topic similarity matrices,
- V and V_k are the n×r and n×k document-topic similarity matrices,
- $A \times A^T$ and $A_k \times A_k^T$ are the m×m term-term similarity matrices,
- $A^T \times A$ and $A_k^T \times A_k$ are the n×n document-document similarity matrices



mapping of m×1 vectors into latent-topic space: $d_j \mapsto U_k^T \times d_j =: d_j'$

$q \mapsto U_k^T \times q =: q'$

scalar-product similarity in latent-topic space: $d_j{}'^T \times q' = ((\Delta_k V_k^T)_{*j})^T \times q'$

LSA in some words (suite)

LSI steps : resume

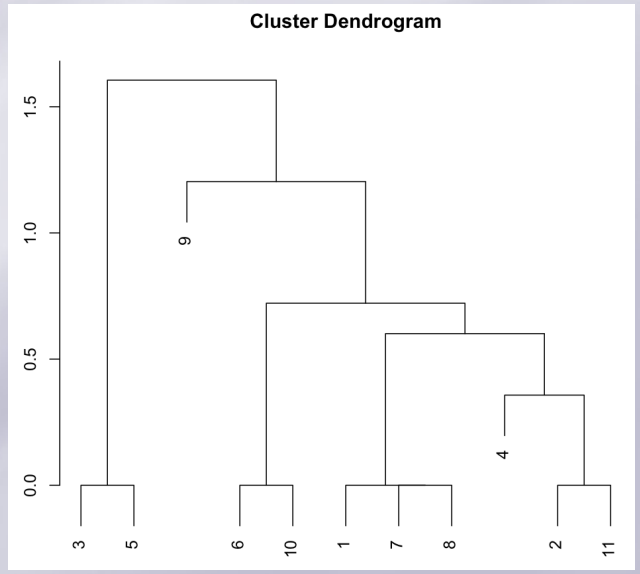
To rank documents with LSI, we just need to :

1. compute weights using a specific term weight scoring system.
2. construct the term-document matrix A.
3. decompose A, compute and truncate all required matrices.
4. find new coordinates of query and document vectors in the reduced k-dimensional space.
5. sort documents in decreasing order of e.g. cosine similarity values.

tf or Tfldf & other clustering methods

Some words about *tfidf* instead of *tf* we used.

- There are very few (11) data. Thus, different weights and distances stay close.
- Ex : a Hierarchical agglomerative clustering (*Ward* method) based on the Euclidian distance-matrix of *tfidf* weights gives (order numbers = ranks at right):



Terms	Term Vector Coordinates	
a	-0.4201	0.0748
arrived	-0.2995	-0.2001
damaged	-0.1206	0.2749
delivery	-0.1576	-0.3046
fire	-0.1206	0.2749
gold	-0.2626	0.3794
in	-0.4201	0.0748
of	-0.4201	0.0748
shipment	-0.2626	0.3794
silver	-0.3151	-0.6093
truck	-0.2995	-0.2001

Note on Polysemy and Synonymity

- A main disadvantage of all term vector models (LSA/LSI) is that terms are assumed to be independent (and termes normally distributed).
- But often this is not the case and Terms can be related by :

1. **Polysemy**; i.e., different meanings in different contexts (e.g. *driving a car* $\Leftrightarrow$ *driving results*).
   - ➛ Some irrelevant documents may have high similarities because they may share some words from the query.
   - ➥ This affects **precision**.
2. **Synonymy**; the same meaning (e.g. *car insurance* $\Leftrightarrow$ *auto insurance*).
   - ➛ The similarity of some relevant documents with the query can be low just because they do not share the same terms.
   - ➥ This affects **recall**.

Gaps of LSA/LSI

Cons :

- LSA measures the co-occurrence of words
- LSA is purely verbal, how about other "vocabulary"
- LSA vectors are context-free
- Word context is not considered
- LSA neglects word order
- The direct relation between Latent semantic space and SVD are contested

. . . ⇝

Gaps of LSA/LSI (suite)

Pros :

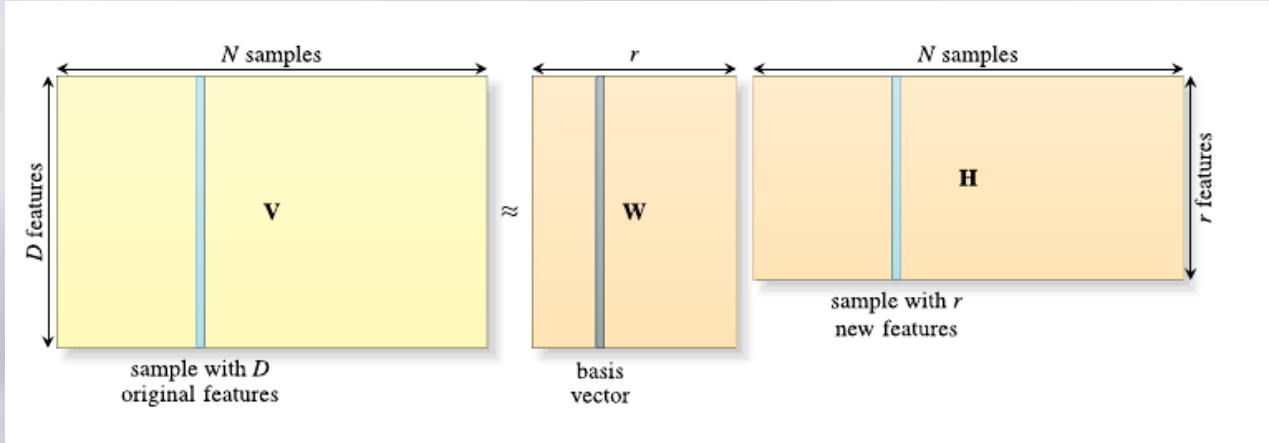
- The truncated SVD in one sense captures most of the important underlying structure in the association of terms and documents
- At the same time removes the noise or variability in word usage that plagues word-based retrieval methods
- Since the number of dimensions k is much smaller than the number of unique terms (m), minor differences in terminology will be ignored
- Terms which occur in similar documents will be near to each other in the k-dimensional vector space even if they never co-occur in the same document
- Also some documents which do not share any words with a users query may none the less be near to it in k-space.

➛ Towards PLSA ...

Non-negative matrix Factorization

Another tool for dimensionality reduction

- Decomposes a matrix (containing only non-negative coefficients) into the product of two other matrices (also composed of non-negative coefficients):
  - a parts-based matrix (W) and
  - a matrix (H) containing the fractional combinations of the parts that approximate the original data.

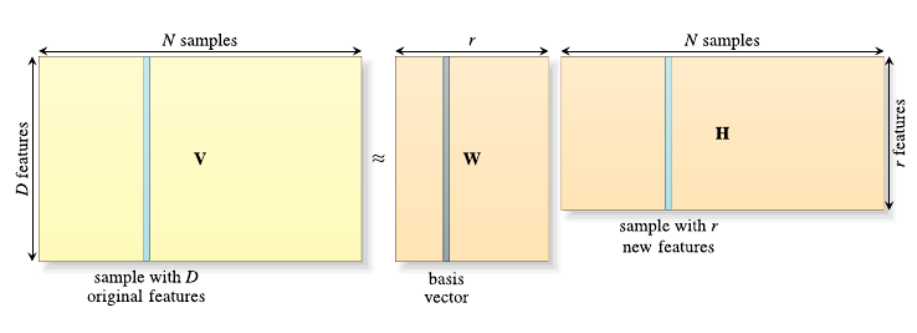


Non-negative matrix Factorization (suite)

- Each column of the original matrix V contains a sample with D features,

- NMF is a method for extracting a new set of r **features** from the original data.

- The parts-based matrix, W will contain the *basis vectors* (one per column), which define a new set of features as an additive function of the original ones.



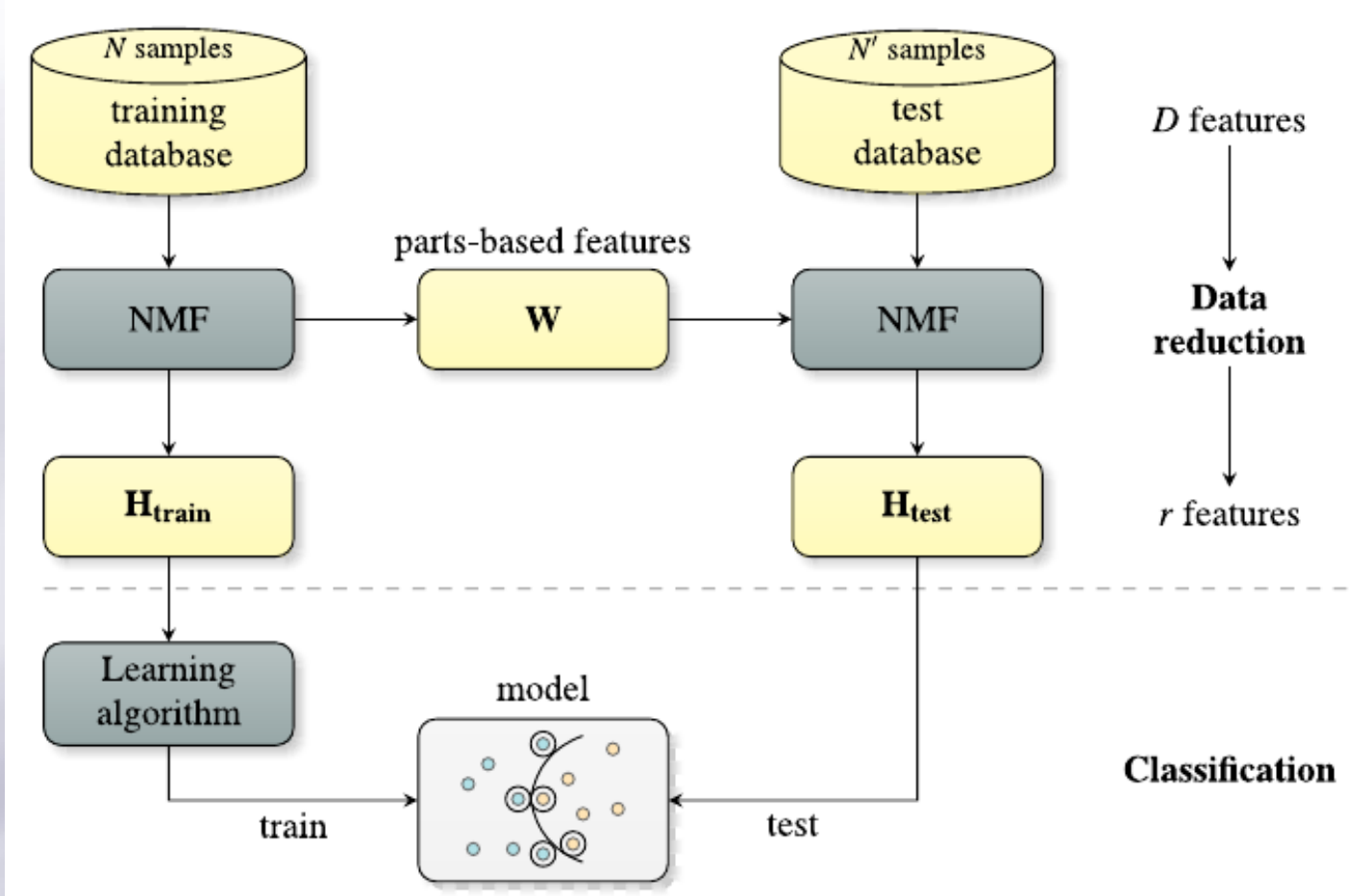
- H will contain the fractional combination of basis vectors that is used to create an approximation of the original samples in V.

- Each column of H will contain a sample (mapped to a new r-dimensional space), which results from superposing the (fractional) contribution of each individual basis vector that approximates the original sample data

- The original data is reconstructed through additive combinations of the parts-based factorized matrix representation.

Non-negative matrix Factorization (suite)

Using NMF :



Word2vect

- Word2vect=un modèle de Réseau de neurones (NN) *skip-gram*
- Un NN assez simple avec une seule couche cachée
- Mais on entraîne ce NN non pas pour ce qu'il a appris mais juste pour extraire les pondérations de sa couche cachée !
  - ➛ Ces pondérations sont justement les vecteurs de mots ("word vector") que l'on cherche à apprendre !
- Cette une technique connue utilisée dans l'apprentissage non-supervisé des *features*.
  - ➛ On utilise le NN pour compresser un vecteur d'entrée dans la couche cachée et de le décompresser vers la couche de sortie.
    - ☞ Idées des CNN...
- Les NN et le sens des couches caches (Hidden layers) : expliquer !

Word2vect (suite)

- Après la phase d'apprentissage, on délaisse le output layer Hidden → l'étape représente ici décompression) et on utilise seulement le hidden layer.
  - ➢ Cette technique est utilisée par exemple pour reconnaître les *features* des images sans les avoir au préalable annotées.

☞ L'unique couche couchée du NN représente un espace corrélé à Input.
A la fin de l'apprentissage, on a (en théorie) trouvé une corrélation entre hidden et output
→ On n'a pas besoin de l'output; on peut bien utiliser le hidden.
Notons que le hidden est ici de taille moindre car on veut réduire la dimension.
Mais cela peut être l'inverse : on peut augmenter la dimension (dans d'autres buts).
On peut aussi avoir 2 couches cachées : meilleure corrélations ?

Word2vect (suite)

- L'entraînement du NN fonctionne de la manière suivante : Étant donné un mot "centrale" w_c (input word pour le NN) d'une phrase, on prend aléatoirement un mot w_v au "voisinage" de w_c et le NN nous dira la probabilité pour N'IMPORTE quel mot du vocabulaire d'être le w_v que nous avons choisi.

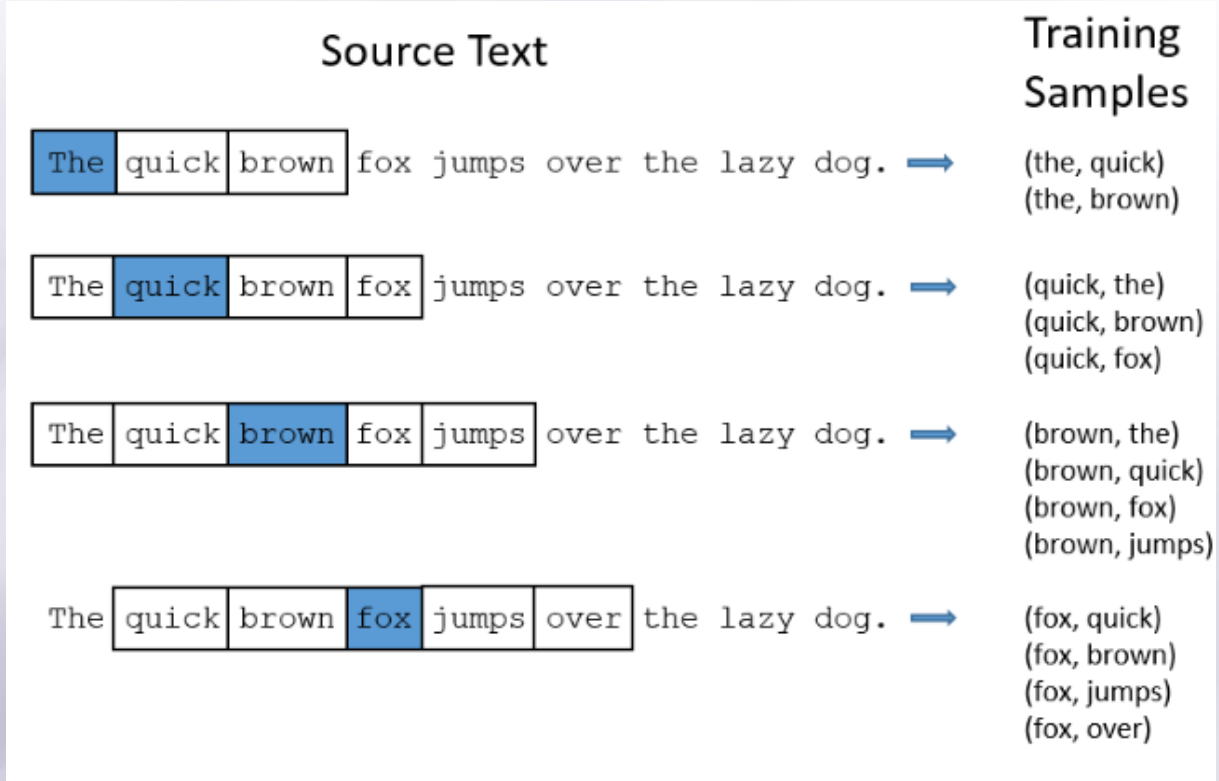
  ➛"voisinage" : les mots dans une fenêtre d'une certaine taille (dans la pratique, 5 mots voisins à gauche et 5 à droite) autour de w_c.

  Par exemple : si le NN est entraîné pour un input = "Bordeaux", les probas en sortie seront plus élevées pour les mots tels que "Gironde" et "Médoc" par rapport aux mots sans rapport tels que "Choucroute" ou "Kangourou" (qui ne se seraient pas trouvés avec "Bordeaux" ou avec "Médoc" ou "Gironde" dans des phrases.)

- Pour cela, le NN est entraîné par des paires de mots.

Word2vect (suite)

Exemple : p our la phrase "The quick brown fox jumps over the lazy dog." et une fenêtre de taille 2, on aura les paires



☞ Notons les paires de mots.

Word2vect (suite)

• En fait, le NN va apprendre des statistiques de combien et comment (contexte) ces paires ont figuré ensemble.

➛ Dans notre exemple, le NN aura croisé souvent la paire ("Bordeaux", "Gironde") que ("Bordeaux", "Sasquatch").
La probabilité de "Gironde" sera plus élevée que pour "Sasquatch" de se trouver au voisinage de "Bordeaux".

Comment présenter les mots au NN :

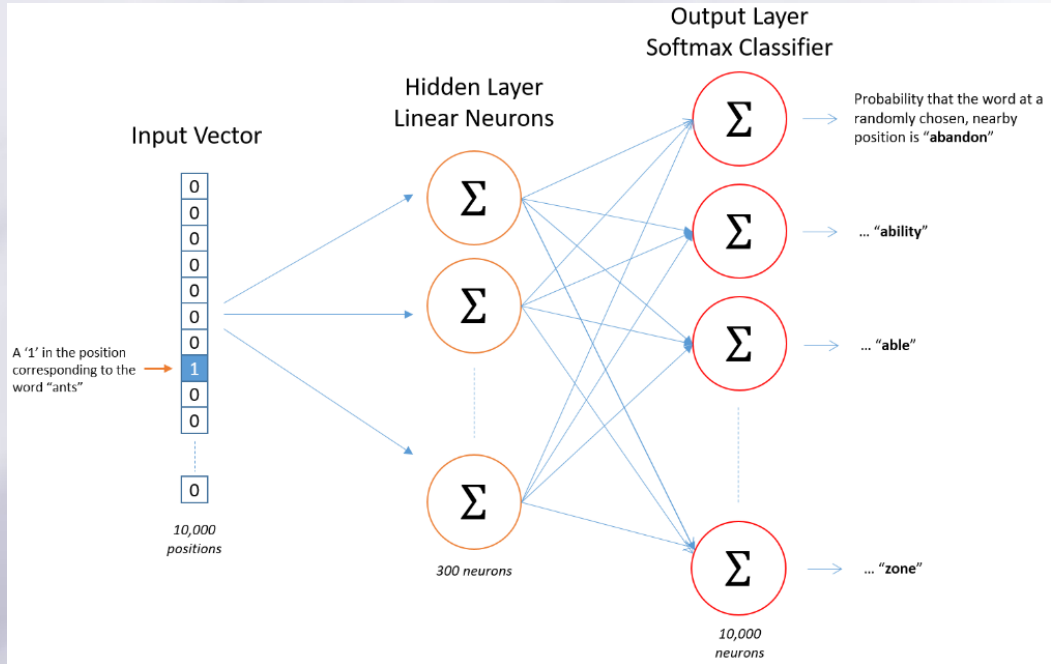
• Supposons avoir un vocabulaire de taille 10000 (mots uniques, sans *stemming* pour simplifier) dont P; ex. le mot "ants".

• Les mots numérisés par One-hot coding : un vecteur par mot

➛ ce vecteur aura 10000 composants (un par mot dans le vocabulaire) avec un "1" dans la case correspondant à l'indexe de "ants" (mots triés dans le vocabulaire) et 0 ailleurs (9999 zéros).

Word2vect (suite)

- La sortie du NN est un seul vecteur de 10000 positions contenant, pour chaque mot w du vocabulaire, la proba pour que un mot aléatoire du vocabulaire proche de "ants" soit w.



➛ Les 300 neurones de Hidden : voir plus bas.

Word2vect (suite)

Fonctionnement du NN

- Les neurones de la couche cachée n'ont pas de fonction d'activation.
- La couche de sortie utilise la fonction **Softmax**.

- Pour l'entraînement : on prend une paire de mots,
  - ➛ Les deux mots encodés par One-hot coding.

On présente le premier en input, le 2e en sortie (le mot attendu).
Le NN affiche des probas (cf. SoftMax) et non une sortie one-hot, et dans le cas idéal, il faudrait avoir une proba de "1" dans la même case du vecteur du mot attendu.

☞ N.B. : On voit ici pourquoi on a une représentation one-hot des mots :
Si on utilise un codage (p. ex.) *grey*, comment rattacher la probabilité au mot attendu ?

Word2vect (suite)

La couche cachée :

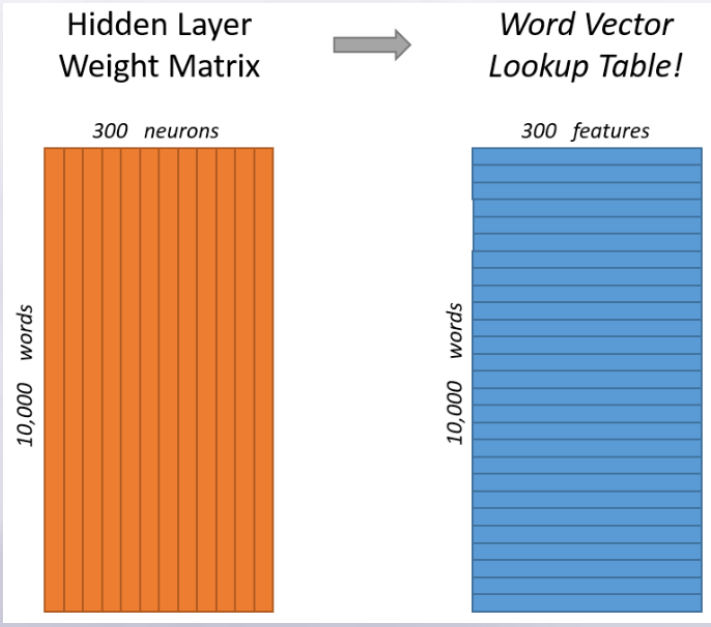
- Supposons décider d'avoir des vecteurs de 300 composants (de 10000, on passe à 300 : compression).
- Dans ce cas, le Hidden sera une matrice de pondérations de 10000×300.
  - ➛ 10000 : une ligne de la matrice par mot du vocabulaire.
- In fine, à la fin de l'apprentissage, la ligne correspondant au mot en Input qui nous intéressera.
  - ➛ On note la réduction de dimension : de 10000 à 300.
    - ➛ La valeur 300 est celle que Goggle a utilisé pour son modèle entraîné sur "Google news dataset" (voir https://code.google.com/archive/p/word2vec/).
- Pour un cas donné, on peaufine ce paramètre.

Word2vect (suite)

- ☞ En fait, ce sont les lignes de cette matrice de pondération qui nous intéressent (non pas la somme faite sur les neurones du hidden)
  - ➛ Cette somme est faite dans les neurones de Hidden où on a que 300 sommes (dont 299 à zéros) :
    Oor il nous faut un vecteur par mot et non une valeur d'où la ligne de la matrice.

Word2vect (suite)

- Si on regarde cette matrice de pondération



➛ La matrice de pondérations (entre Input et Hidden) se comporte comme une table et chacune de ses lignes est justement le "vecteur du mot" pour le mot présenté en Input.

- Le but final de cet apprentissage est seulement la matrice de gauche.

Le output est délaissée à la fin.

Word2vect (suite)

Le pourquoi du codage one-hot

- A priori, il y a beaucoup de zéros et on pourrait faire plus efficace.
- Si on multiplie un vecteur one-hot de taille 100000 par une matrice de 100000×300, on choisit effectivement la ligne de la matrice correspondant au "1" dans le vecteur. On peut le voir dans cette figure :

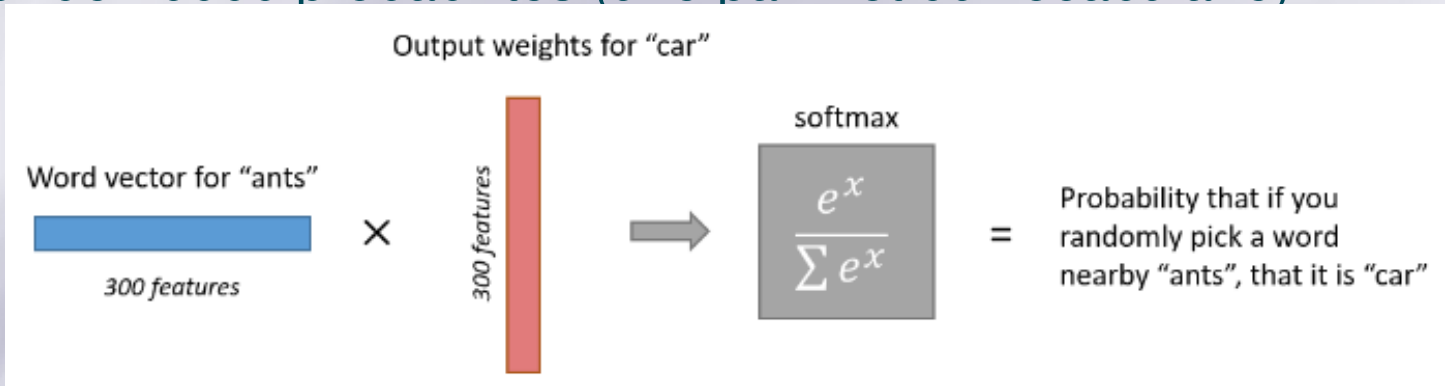
$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

Word2vect (suite)

La couche output :

- De l'exemple ci-dessus, le vecteur 1 x 300 pour "ants" arrive à output. Softmax produira ensuite une valeur entre 0..1 (et la somme de toutes ces valeurs = 1 –> probas).
- Exemple de calcul pour le mot "car" :

le "vecteur de mot" récupéré sur le Hidden (de W1 = des neurones du Hidden, voir ci-dessus) est multiplié par les pondérations W2 (entre Hidden et Output) puis la somme est passé à Softmax pour produire un vecteur de 10000 probabilités (une par mot du vocabulaire).



Word2vect (suite)

- Le NN ne sait rien des relations entre la proba de sortie et le mot en input (car la proba de sortie concerne le 2e mot de la paire).
- Le NN n'apprend pas non plus différentes proba pour les mots avant ou après le mot en entrée.
- Notons aussi que même si le mot 'york' est toujours précédé de 'new' (on le sait à 100% sur notre corpus que 'New' est dans le voisinage de 'York'), si on choisissait 10 mots dans le voisinage de 'York' et on en prend un, la proba que ce mot soit 'new' n'est pas 100% car on a peut choisi un autre mot (que 'New').
- Si deux mots ont un contexte similaire (se trouvent au voisinage l'un l'autre), le NN doit calculer des résultats similaires pour ces deux mots.
  - ➛ C'est à dire : deux "vecteurs de mots" similaires.
  - ➛ Par exemple, "Intelligent" et "smart" ont des contextes similaires. De même pour "engine" et "transmission".
- On peut dire que le NN fait un travail du stemming dans la mesure où il trouve les mêmes contextes pour "ants" et "ant".

Word2vect (suite)

Remarques et optimisation :

- Les matrices sont grandes et il faut des moyens d'optimisation.

1- **Collage** : on passe sur les mots pour coller ensemble par exemple "new" et "york" en en faire un seule mot "New_York". On peut repasser et obtenir par exemple "new_york_city".

2- **Subsampling** : on évite des paires comme "fox" et "the" (de la phrase utilisée ci-dessus) car "the" ne nous apprend rien sur "fox".

3- En ajustant le sampling rate, on choisit mieux les mots du vocabulaire. Cela ressembler à éliminer des mots **moins fréquents**.

4 - **Négative sampling** : pour la rétro-propagation, au lieu de propager tous les zéros en output, on en utilisera très peu

- ➛ (on dit "négative" car "0"=absence).
- ➛ La valeur par défaut utilisée est 5 : seuls 5 des zéros du output seront utilisés pour mettre à jours les pondérations.

Word2vect (suite)

Addendum : à propos de Softmax

Softmax (ou la fonction exponentielle normalisée) :

Cette fonction est une généralisation de la fonction logistique donne l'image d'un vecteur de réels vers $[0, 1]$

$$f_i(\overrightarrow{x}) = \frac{e^x}{\sum_{j=1}^{J} e^{x_j}}, \ pour\ i = 1..J \text{ où } J = \text{nombre de neurones de la couche.}$$

L'étendu de Softmax est $[0, 1]$

Cette fonction calcule donc une probabilité pour un neurone parmi les k neurones d'une couche (ici, output).

La dérivée (pour le gradient) de *Softmax* sera :

$$\frac{\partial f_i(\overrightarrow{x})}{\partial x_j} = f_i(\overrightarrow{x})(\delta_{ij} - f_j(\overrightarrow{x})) \text{ où } \delta_{ij} = 1 \ si\ i = j, \ sinon\ 0$$

N.B. : lorsqu'il s'agira d'une probabilité (comme dans notre cas), par exemple pour prédire la classe (j d'un élément (représenté par un vecteur x) et un vecteur de pondération w, on notera :

$$P(classe = j|x) = \frac{e^{x^T \cdot w_j}}{\sum_{k=1}^{K} e^{x^T \cdot w_k}} \text{ pour K} = \text{nombre de classes.}$$

Word2vect (suite)

Un exemple de calcul de Softmax :

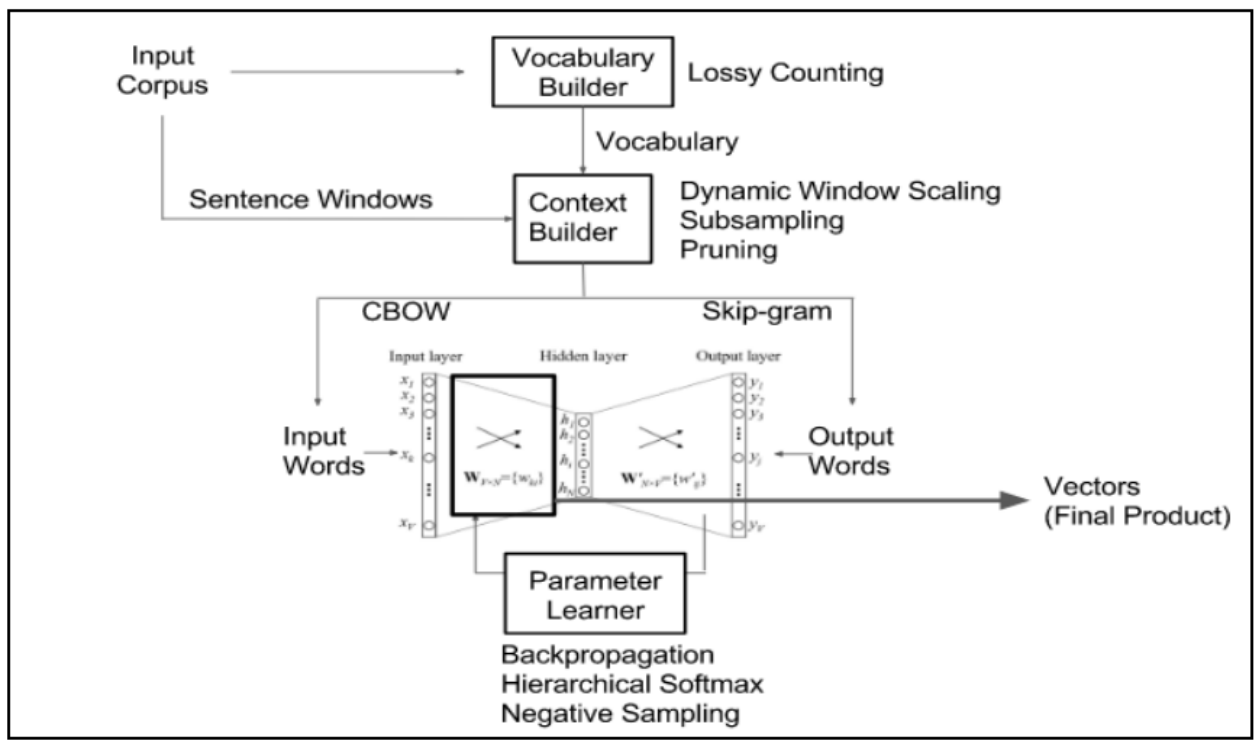
```
import math
z = [1.0, 2.0, 3.0, 4.0, 1.0, 2.0, 3.0]
exp_z = [math.exp(i) for i in z]
print([round(i, 2) for i in exp_z])
# [2.72, 7.39, 20.09, 54.6, 2.72, 7.39, 20.09]

somme_exp_z = sum(exp_z)
print(round(somme_exp_z, 2))
# 114.98

softmax = [round(i / somme_exp_z, 3) for i in exp_z]
print(softmax)
# [0.024, 0.064, 0.175, 0.475, 0.024, 0.064, 0.175]
```

Word2vect (suite)

- Il n'y a pas de CNN
- Word2vec building block (Image credit: Xin Rong)



Towards PLSA

Important (from Grossman and Frieder) :

The key to similarity is not that two terms happen to occur in the same document:

*it is that the two terms appear in the same **CONTEXT***

that is they have very similar neighboring terms .

- The idea from pLSA :

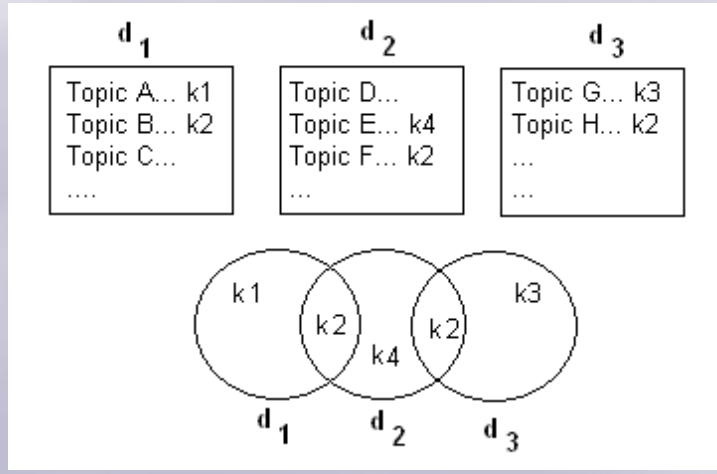
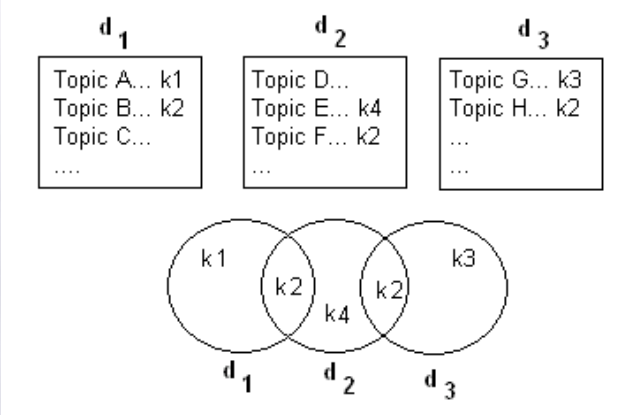


Figure 6: Documents and different Topics of terms

Towards PLSA (suite)



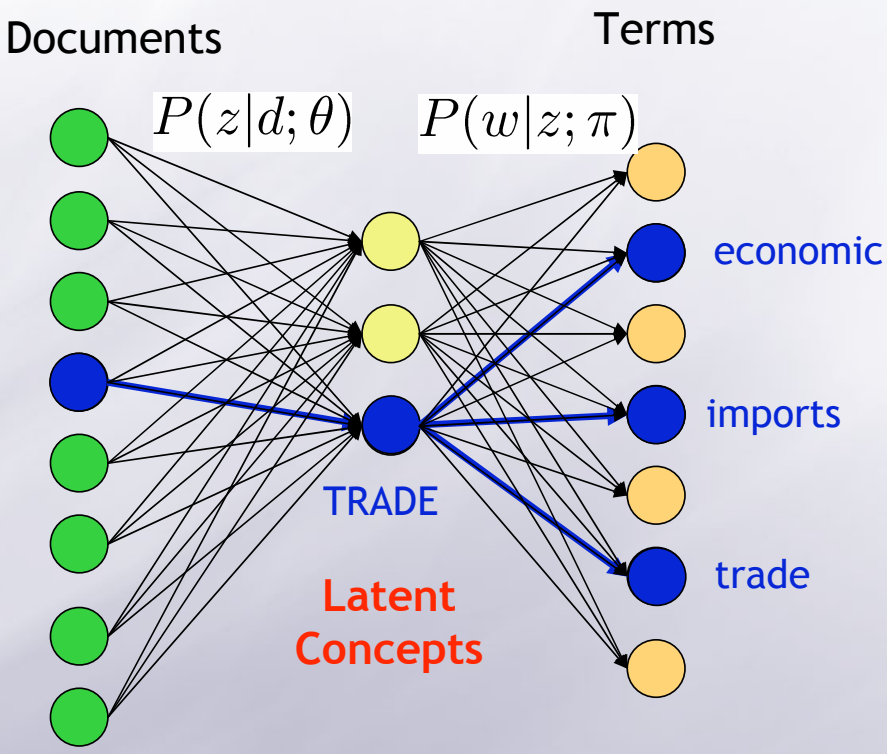
- 1st order term-trem co-occ relation between d_1 and d_2
- 1st order term-trem co-occ relation between d_2 and d_3
- **2nd order** term-trem co-occ relation between d_1 and d_3

Here, the LSI similarity scores based on word counts do not reflect the actual relationship between terms.

- Add polysems and the scenario worsens.

PLSA

PLSA :



☞ Put **image** in place of *Document* and **Visual term** instead of *Terms* and you are in the image world !

PLSA (suite)

- Concept expression probabilities are estimated based on all documents that are dealing with a concept.

- "Unmixing" of superimposed concepts is achieved by statistical learning algorithm.

- No prior knowledge about concepts required :
  ➛ context and term co-occurrences are exploited.

- Latent concepts disappear (generative / Discriminative) at the end.

PLSA (a practical introduction)

- Proposed by Hofmann.
- Starts with the raw term-counts in a given document collection :
  - ➛ Uses the co-occurrence matrix in which each entry $\mathrm{n}(\mathrm{d}, w)$ denotes the frequency of **term** w in **document** d
    - ➛ (d, w) is a *binary event* when data observed.
    - ➛ $\mathrm{n}(\mathrm{d}, w)$ is the number of times this event occurs is the observed data (caused by a *hiden categorial logic*).
- N.B. : note the subtle distinction between this model and the multinomial model.
  - ➛ Here the total event count over all (d, w) is set in advance, and the (d, w) events must apportion the total count.
  - ➛ i.e. (unlike the multinomial model), the corpus must be fixed in advance and that analyzing a new Doc. from outside the corpus takes some special steps.

PLSA (a practical introduction) (suite)

What's called *Topic / Concept / Aspect = Cluster* :

- If you're an author, you may starts composing a document by inducing a probability distribution $\Pr(z)$ over **topics** (clusters) :
  - ➛ you may set a probability of 0.3 for using terms about politics and 0.7 for green-petroleum ones.
  - ➛ Different clusters **cause** events (d, w) with different probabilities.
  - ➛ The joint probability : $\Pr(\mathrm{d}, w) = \sum_{\mathrm{c}} \Pr(z)\ \Pr(\mathrm{d}, w|z) \quad (\mathrm{I})$
- **The BIG hypothesis** :
  assume conditional independence between d and w given c :

$$\Pr(\mathrm{d}, w) = \sum_{z} \Pr(z)\Pr(\mathrm{d}|z)\Pr(w|z) \qquad (\mathrm{II})$$

PLSA (a practical introduction) (suite)

- We had (in II) : $\Pr(d, w) = \sum_z \Pr(z)\Pr(d|z)\Pr(w|z)$
- Retain these important parameters : $\Pr(z), \Pr(d|z), \Pr(w|z)$.
- To estimates them, we use an EM-like procedure :
  - ➛ the E-step parameter $\Pr(z|d, w)$ may be interpreted as a grade of evidence (as in general Bayes $\Pr(H|E)$) that event (d, w) was caused by topic z.

- Let $\Pr(z|d, w) = \dfrac{\Pr(z, d, w)}{\Pr(d, w)}$

$$= \frac{\Pr(z)\Pr(d, w|z)}{\sum_\gamma \Pr(\gamma, d, w)} \quad \text{(take II for the denominator)}$$

PLSA (a practical introduction) (suite)

Recall : $\Pr(z|\mathrm{d},w) = \dfrac{\Pr(z)\Pr(\mathrm{d},w|z)}{\sum_{\gamma}\Pr(\gamma,\mathrm{d},w)}$

- Gives the E-step of EM (remember II) and the big Hyp. :

$$\Pr(z|\mathrm{d},w) = \frac{\Pr(z)\Pr(\mathrm{d}|z)\Pr(w|z)}{\sum_{\gamma}\Pr(\gamma)\Pr(\mathrm{d}|\gamma)\Pr(w|\gamma)} \qquad (\mathrm{III})$$

➛ All computations below will use (III) of the E-step.

➛ The co-occ matrice we give the following ESTIMATIONS :

PLSA (a practical introduction) (suite)

- The M-step of EM (will be used by the next-E-step):

$$\Pr(z) = \frac{\sum_{d,w} n(d,w)\Pr(z|d,w)}{\sum_{\gamma}\sum_{d,w} n(d,w)\Pr(\gamma|d,w)} \qquad (IV)$$

$$\Pr(d|z) = \frac{\sum_{w} n(d,w)\Pr(z|d,w)}{\sum_{d}\sum_{w} n(d,w)\Pr(z|d,w)} \qquad (V)$$

$$\Pr(w|z) = \frac{\sum_{d} n(d,w)\Pr(z|d,w)}{\sum_{w}\sum_{d} n(d,w)\Pr(z|d,w)} \qquad (VI)$$

PLSA (a practical introduction) (suite)

HALT(coffee break) !

- The E-step estimates (III) : $\Pr(z|\mathrm{d}, w)$
- The M-step uses this result to estimate $\Pr(z), \Pr(\mathrm{d}|z), \Pr(w|z)$ which in turn are used by the E-step .... until stabilisation.

Let's note :

- As in EM, the user has to fix *a priori* the number of clusters (arbitrary or by Hold-Out-XV).
- The number of clusters is akin to the number of singular values retained in the LSI (SVD) decomposition;

  But one may use hold-out data for XV to determine it.
- Hofmann also described an enhanced EM procedure.

PLSA (a practical introduction) (suite)

PLSA space :

➛ $p(w|d) = \sum \Pr(w|z)P(z|d)$?

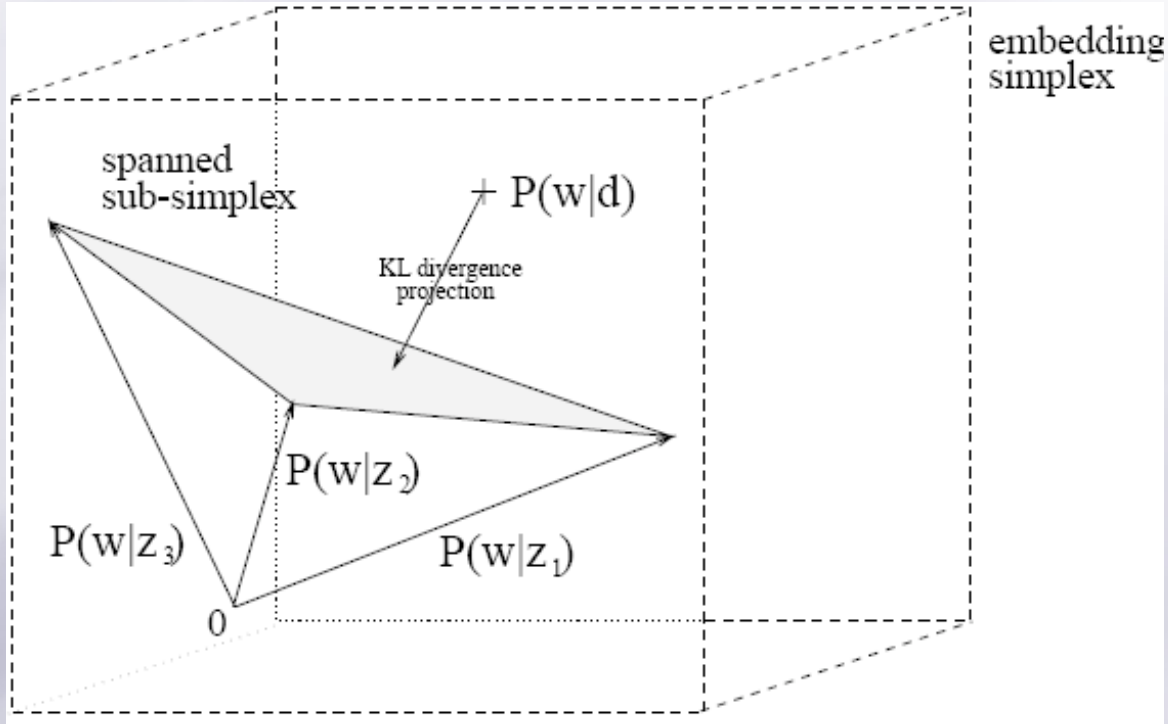


Figure 7: T. Hofmann, Probabilistic Latent Semantic Analysis, UAI 1999.

PLSA (a practical introduction) (suite)

Using the Topic model (as PLSI) on a corpus :

- First, compute the set of four parameters estimated above (III, IV, V, VI) on a corpus.
- Then, for each document d and each cluster z, pre-compute :

$$\Pr(z|d) = \frac{\Pr(z)\Pr(d|z)}{\sum_{\gamma} \Pr(\gamma)\Pr(d|\gamma)} \qquad (VII)$$

every thing on the right-hand side is estimated parameters.

PLSA (a practical introduction) (suite)

- A query q (a bag of words) has to be **folded** into the system. **Freeze** the pre calculated parameters
  new params $\forall z, w \ \Pr(z|q,w)$ and $\forall z \Pr(q|z)$ are estimated as :

$$\Pr(z|q,w) = \frac{\Pr(z)\Pr(q|z)\Pr(w|z)}{\sum_{\gamma} \Pr(\gamma)\Pr(q|\gamma)\Pr(w|\gamma)}$$

$$\Pr(q|z) = \frac{\sum_{w} n(q,w)\Pr(z|q,w)}{\sum_{w} n(q,w)\Pr(z|q,w) + \sum_{d}\sum_{w} n(d,w)\Pr(z|d,w)}$$

This is itself an iterative procedure with the coupling shown by the red color.

PLSA (a practical introduction) (suite)

- Once $\Pr(z|q)$ and $\Pr(z|d)$ are known for all d, one may use the vector of posterior class probabilities as a representation, just as the projection via U or V^T is used in LSI.

  ➛ i.e., the similarity between q and d may be defined by e.g.,

$$\sum_z \Pr(z|q)\Pr(z|d), \quad \text{or by} \quad \sum_z \Pr(z)\Pr(d|z)\Pr(q|z),$$

for both of which the similarity uses a dot-product of vectors.

PLSA (a practical introduction) (suite)

Relation to LSA : Mixture decomposition (PLSA) vs. SVD

- Rewrite the aspect model in a matrix notation ($A = USV^T$)
  - ➛ Define matrices by $\hat{U} = (Pr(d_i|z_k))_{i,k}$, $\hat{V} = (Pr(w_j|z_k))_{j,k}$, $\hat{S} = diag(P(z_k))$
- Rewrite then joint probability model P as $P = \hat{U}\hat{S}\hat{V}^T$
  - ➛ $P = \hat{U}\hat{S}\hat{V}^T$: documents / concepts / terms probability
- Comparing both decompositions (linear algebra) :
  1. The weighted sum over outer products between rows of $\hat{U}$ and $\hat{V}$ reflects conditional independence in PLSA.
  2. The left/right eigenvectors in SVD correspond to the factors $P(w|z)$ and to the component distributions $P(d|z)$
  3. The mixing proportions $P(z)$ in PLSA substitute the singular values of the SVD in LSA.

PLSA (a practical introduction) (suite)

Another fundamental difference between PLSA and LSA :

• (remember from PCA) the objective function utilized to determine the optimal decomposition approximation (explicit maximization of the predictive power in PLSA)

➛ In LSA, this is the L2-norm (Frobenius norm)

➥ corresponds to an implicit additive Gaussian noise assumption on counts.

➛ while PLSA relies on the likelihood function of multinomial sampling

Modeling power of PLSA :

the mixture approximation P of the co-occurrence table is a well-defined probability distribution

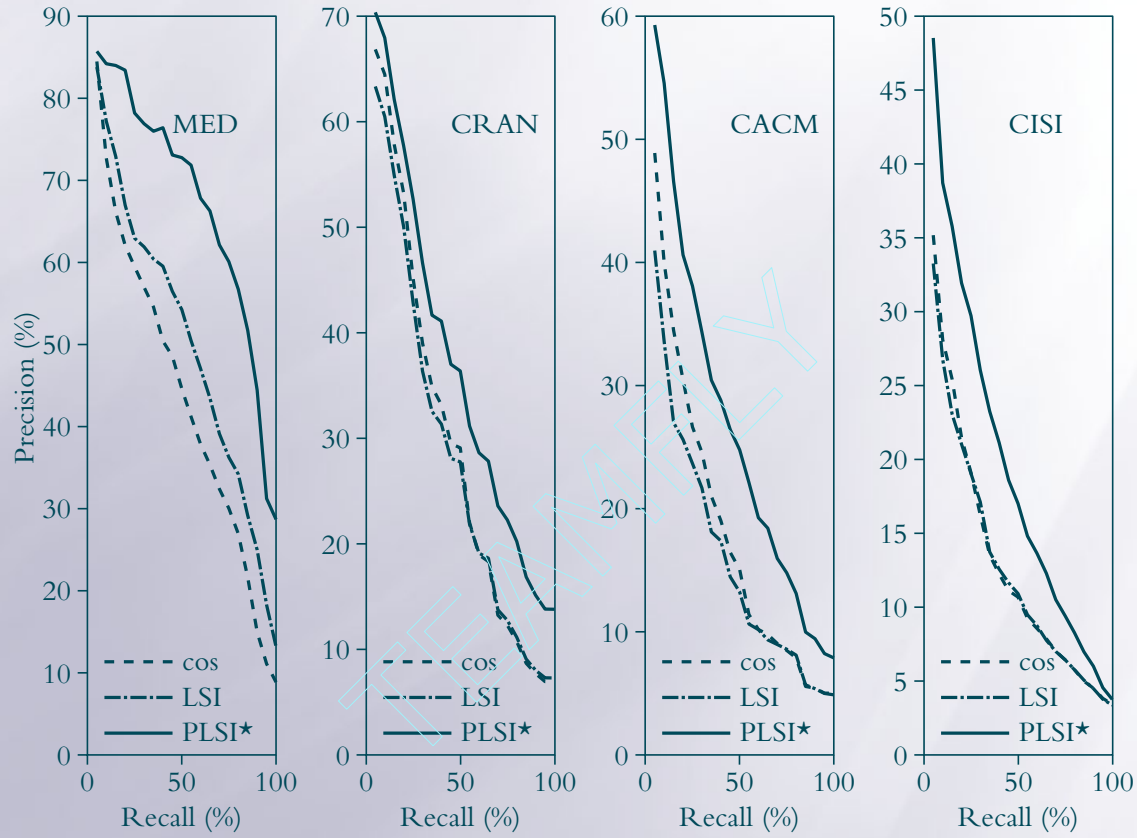
factors have a clear probabilistic meaning in terms of mixture component distributions.

PLSA (a practical introduction) (suite)

PLSA performances :

PLSA compares favorably in terms of recall and precision with standard **TFIDF** cosine-based ranking.

➔ PLSI performances on from 1000 (MED) to 3000 (CACM) corpora



PLSA (suite)

PLSA : Generative model

- Generate a document d with probability $\mathrm{p}(\mathrm{d})$
- Having observed d generate a semantic factor with probability $\mathrm{p}(z|\mathrm{d})$
- Having observed a semantic factor generate a word with probability $\mathrm{p}(w|z)$

➛ z acts as a bottleneck variable in predicting words

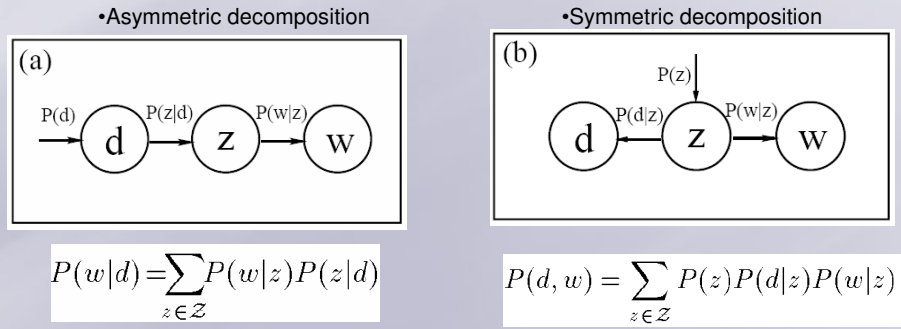


Figure 8: Generative model

PLSA (suite) (suite)

pLSA: Graphical Model

$$P(w|d) = \sum_z P(w|z)P(z|d)$$

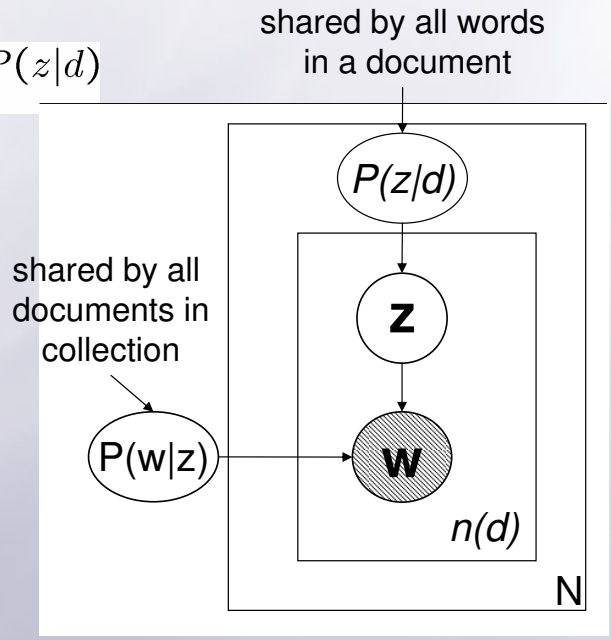


Figure 9: graphical model

PLSA (suite) (suite)

Example (based on 1568 documents and 128 topics), "Segment"

10 most probable words in the respective latent classes (or factors)

"segment 1"	"segment 2"	"matrix 1"	"matrix 2"	"line 1"	"line 2"	"power 1"	power 2"
imag	speaker	robust	manufactur	constraint	alpha	POWER	load
SEGMENT	speech	MATRIX	cell	LINE	redshift	spectrum	memori
texture	recogni	eigenvalu	part	match	LINE	omega	vlsi
color	signal	uncertainti	MATRIX	locat	galaxi	mpc	POWER
tissue	train	plane	cellular	imag	quasar	hsup	systolic
brain	hmm	linear	famili	geometr	absorp	larg	input
slice	source	condition	design	impos	high	redshift	complex
cluster	speakerind.	perturb	machinepart	segment	ssup	galaxi	arrai
mri	SEGMENT	root	format	fundament	densiti	standard	present
volume	sound	suffici	group	recogn	veloc	model	implement

Most relevant latent classes for word "segment"

Most relevant latent classes for word "matrix"

Most relevant latent classes for word "line"

Most relevant latent classes for word "power"

Documents are previously pre-processed by an standard stop-word list and stemmer

PLSA (suite) (suite)

Document 1, $P\{z_k|d_1, w_j = \text{`}segment\text{`}\} = (0.951, 0.0001, \ldots)$
$P\{w_j = \text{`}segment\text{`}|d_1\} = 0.06$

SEGMENT medic imag challeng problem field imag analysi diagnost base proper **SEGMENT** digit imag **SEGMENT** medic imag need applic involv estim boundari object classif tissu abnorm shape analysi contour detec textur **SEGMENT** despit exist techniqu **SEGMENT** specif medic imag remain crucial problem [...]

Document 2, $P\{z_k|d_2, w_j = \text{`}segment\text{`}\} = (0.025, 0.867, \ldots)$
$P\{w_j = \text{`}segment\text{`}|d_2\} = 0.010$

consid signal origin sequenc sourc specif problem **SEGMENT** signal relat **SEGMENT** sourc address issu wide applic field report describ resolu method ergod hidden markov model hmm hmm state correspond signal sourc signal sourc sequenc determin decod procedur viterbi algorithm forward algorithm observ sequenc baumwelch train estim hmm paramet train materi applic multipl signal sourc identif problem experi perform unknown speaker identif [...]

Figure 10: Example (cont.)

PLSA (suite) (suite)

From Sivic et al. ICCV 2005

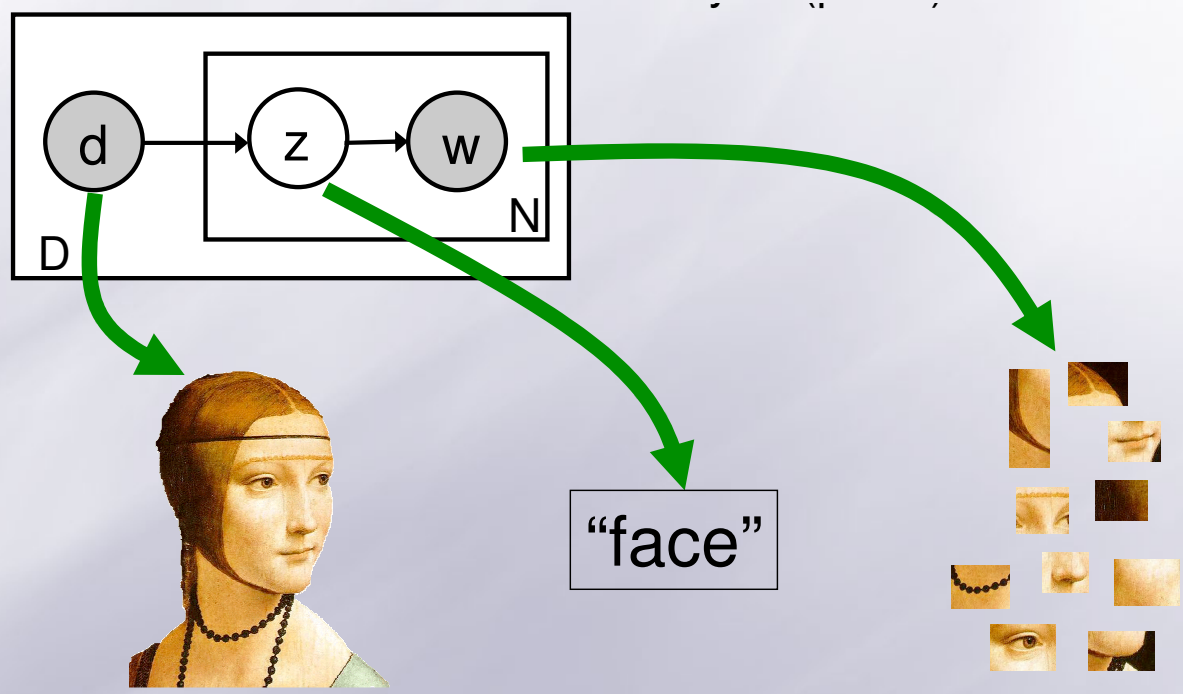


Figure 11: pLSA and Image

LDA

- Problems with pLSA
  - ➛ Not a well-defined generative model of documents d
  - ➛ d is a dummy index into the list of documents in the training set (as many values as documents)
  - ➛ No natural way to assign probability to a previously unseen document
  - ➛ Number of parameters to be estimated grows with size of training set

Recall : pLSA deal with previously unseen documents :

- ➛ Folding-in Heuristic

  First train on Corpus to obtain $P(w_j|Z_k)$

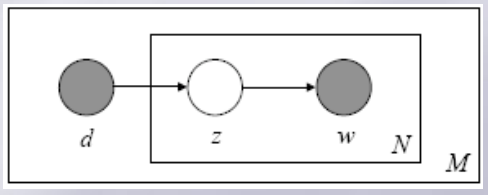
  Now re-run same training EM algorithm, but don't re-estimate $P(w_j|Z_k)$ and let $D = \{d_{unseen}\}$

LDA (suite)

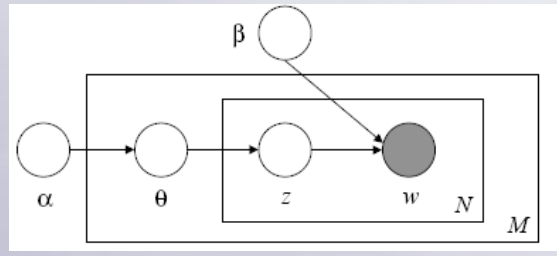
LDA : Latent Dirichlet Allocation treats the topic mixture weights as a k-parameter hidden random variable and places a Dirichlet prior on the multinomial mixing weights

- Dirichlet distribution is conjugate to the multinomial distribution

➥ The most natural prior to choose is Dirichlet and the posterior distribution is also a Dirichlet !



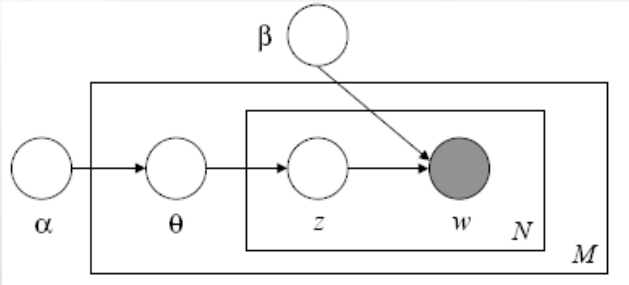
pLSA



LDA

Figure 12: $\alpha = \text{Dirichlet prior}, \Theta : \text{multinomial}$

LDA (suite)

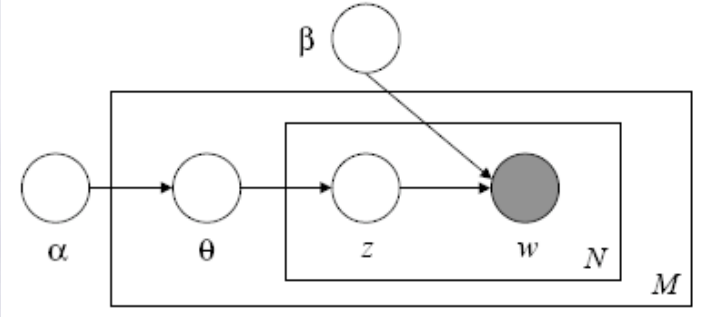


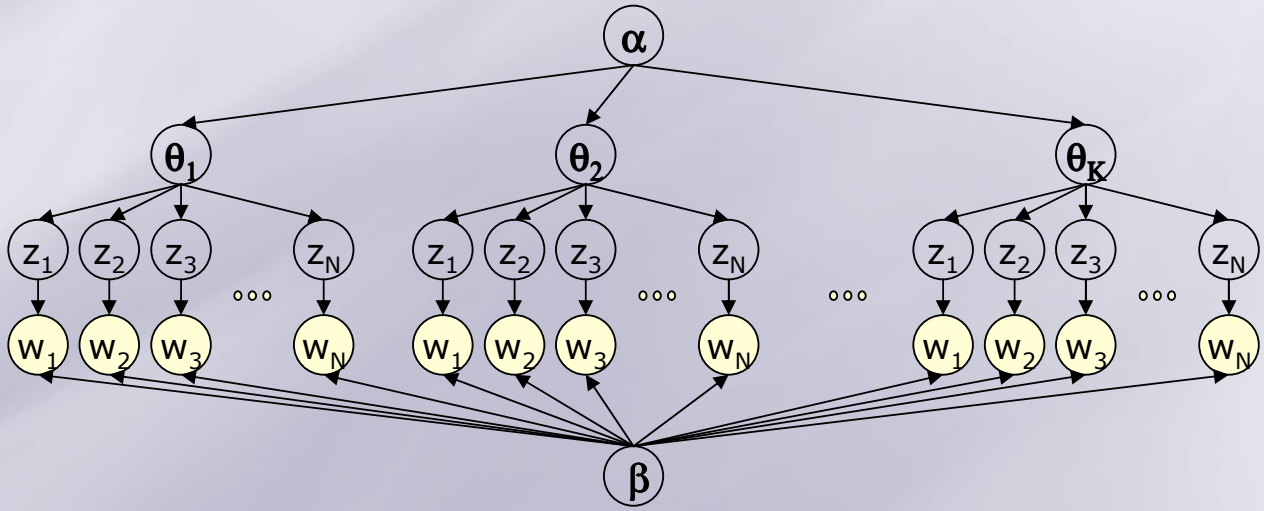
LDA

- α, β are corpus-level documents that are sampled once in the corpus creating generative model
- α, β must be estimated before we can find the topic mixing proportions belonging to a previously unseen document
- Sivic specializes them by $\alpha = \beta = 0.5$

LDA (suite)

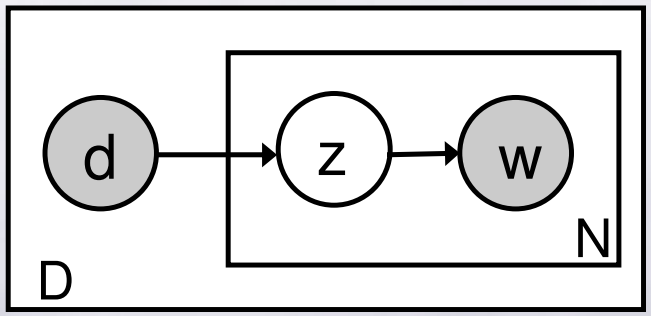
Generative Model (LDA)





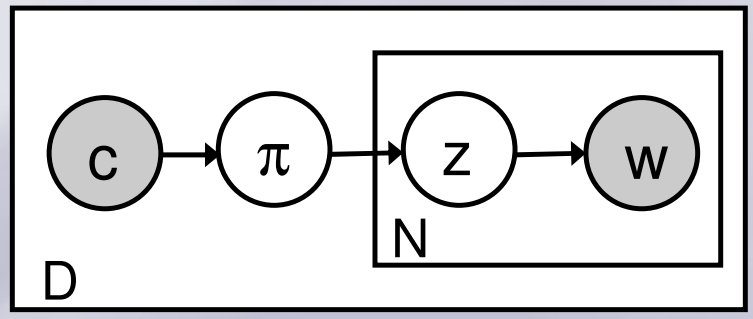
PLSA & LDA in Image

Recall (simplified LDA):



pLSA

Hoffman, 2001



LDA

Blei et al., 2001

PLSA & LDA in Image (suite)

From Sivic et al. ICCV 2005

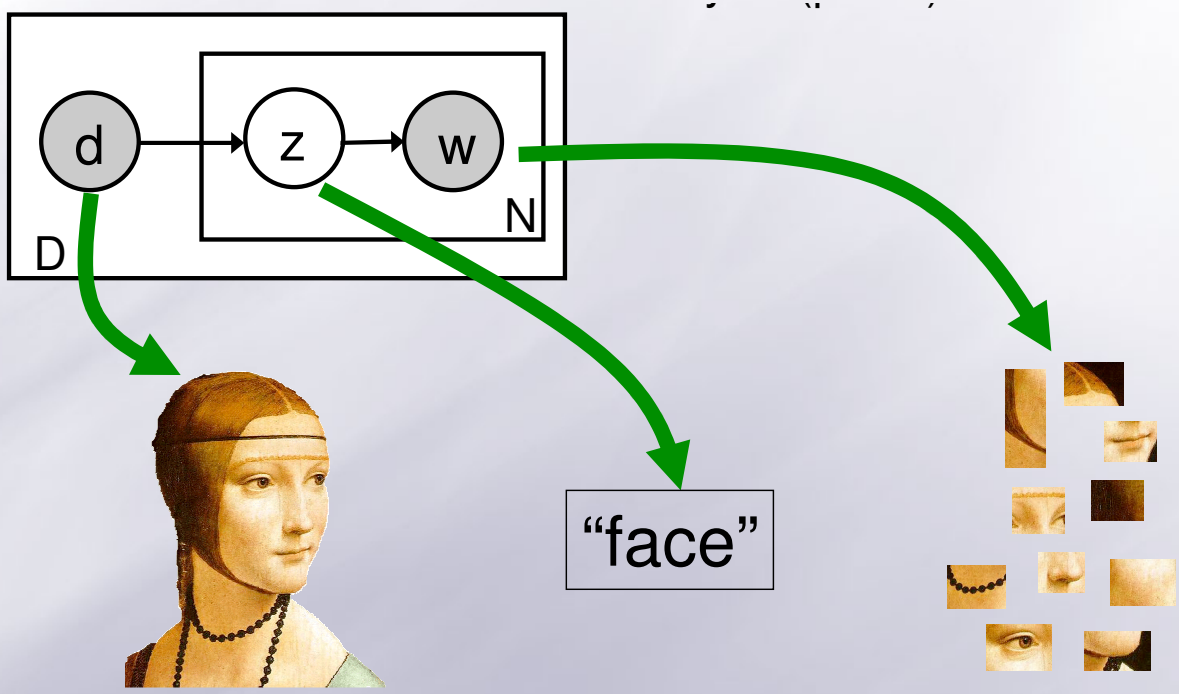


Figure 13: pLSA and Image

PLSA & LDA in Image (suite)

From Fei-Fei et al. ICCV 2005

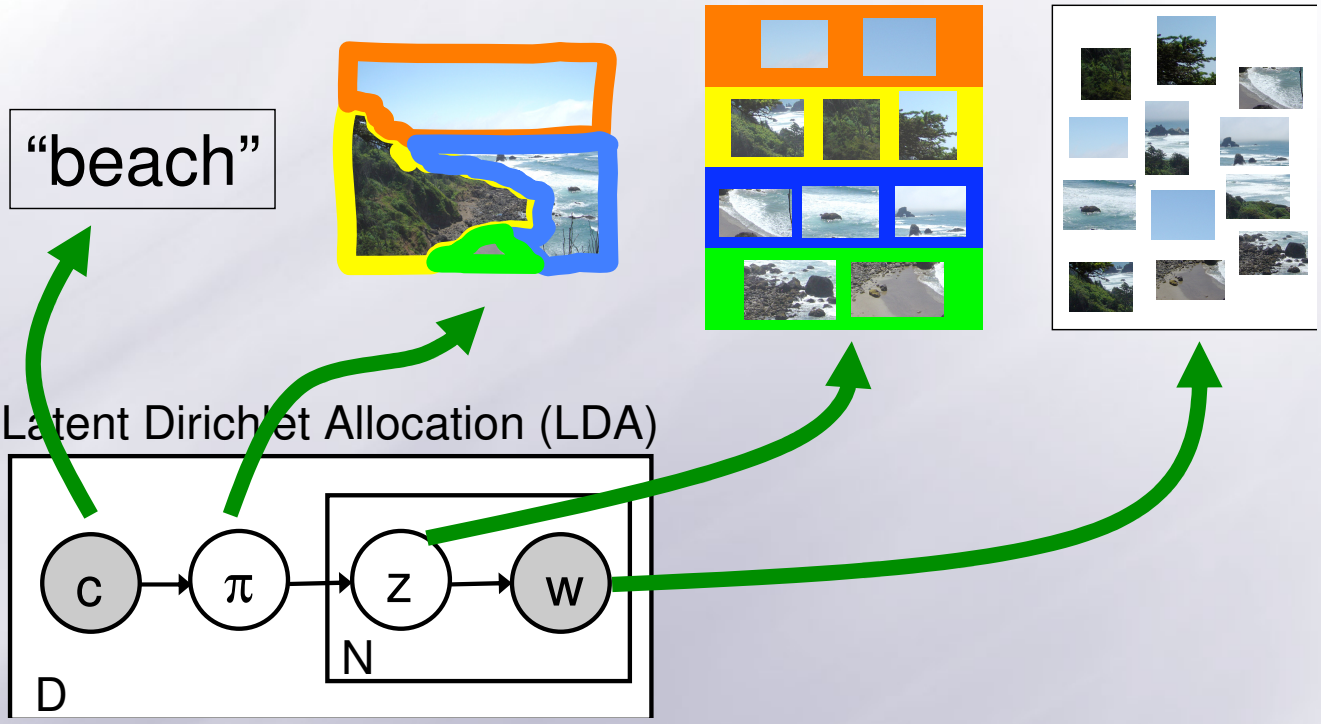
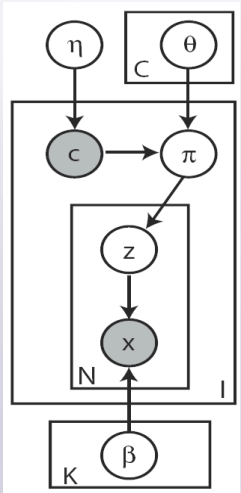


Figure 14: LDA and Image

PLSA & LDA in Image (suite)

LDA From Fei-Fei, Perona : 'A Bayesian Hierarchical Model for Learning Natural Scene Categories'

dashed 'c' : category,
'C' : #categories,
'z' : theme,
dashed 'x' : codeword,
π : probability vector over 'themes',
$K = \#Themes$,
β is matrix of size $K \times T$ (#themes x #words)



PLSA & LDA in Image (suite)

From Fei-Fei, Perona : Flow Chart

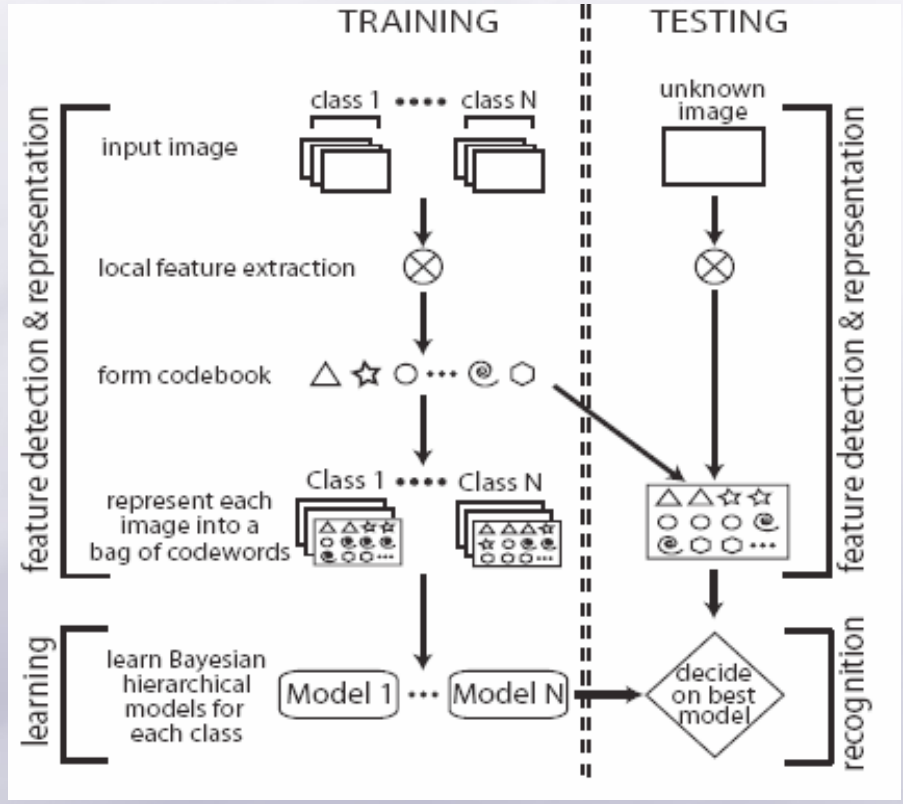


Figure 15: Flow Chart

PLSA & LDA in Image (suite)

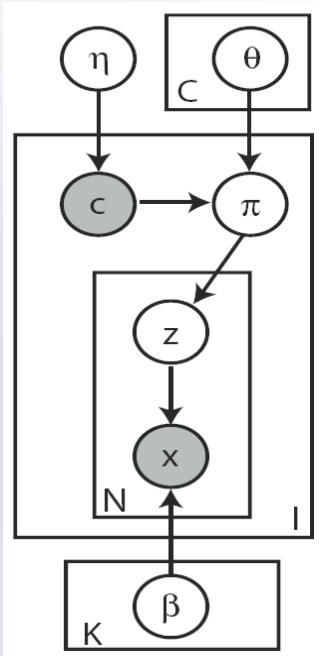
How to Generate an Image ? (more on the next slide)

- Choose a scene (mountain, beach, ...) : a **category** (c)
- Given scene generate an intermediate probability vector over '**themes**' (π)

For each word:

- Determine current theme from mixture of themes (z)
- Draw a codeword from that theme (x)

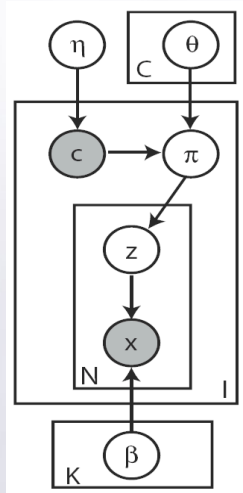
$$P(x, z, \pi, c|\Theta, \eta, \beta) = P(c|\eta)P(\pi|c, \Theta)\prod_{n=1}^{N} P(z_n|\pi)P(x_n|z_n, \beta)$$

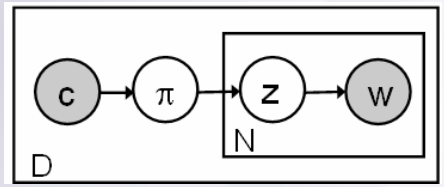


PLSA & LDA in Image (suite)

How to Generate an Image (details) ?

- Choose a category label $c \sim p(c|\eta)$
  η : prior over scene category (multinomial)
- Choose $\pi \sim p(\pi|c, \Theta)$
  π is multinomial distribution over themes Z
- Θ is a $C \times K$ (#category x #themes)
- Θ_k is k-dimensional Dirichlet parameter condition on the category c
- For each of the N patches
  Choose theme $Z_n \sim mult(\pi)$
  Choose patch $X_n \sim p(X_n|Z_n, \beta)$
  β is matrix of size $K \times T$ (#themes x #words)





PLSA & LDA in Image (suite)

How to do inference (decision)

• How to make decision on a novel image (how to get c : category) ?

$$p(c|x, \Theta, \beta, \eta) \propto p(x|c, \Theta, \beta)p(c|\eta) \propto p(x|c, \Theta, \beta)$$

• Integrate over latent variables to get:

$$p(x|c, \Theta, \beta) = \int p(\pi|\Theta, c) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\pi)p(x_n|z_n, \beta) \right) d\pi$$

➛ Not easy to do such Variational Inference

➛ Gibs sampling is supposed to be easier.