Text Processing on the Web

Week 8
Text Summarization

The material for these slides are largely taken from the ACL tutorial by Daniel Marcu and Eduard Hovy of ISI

Recap: Question Answering

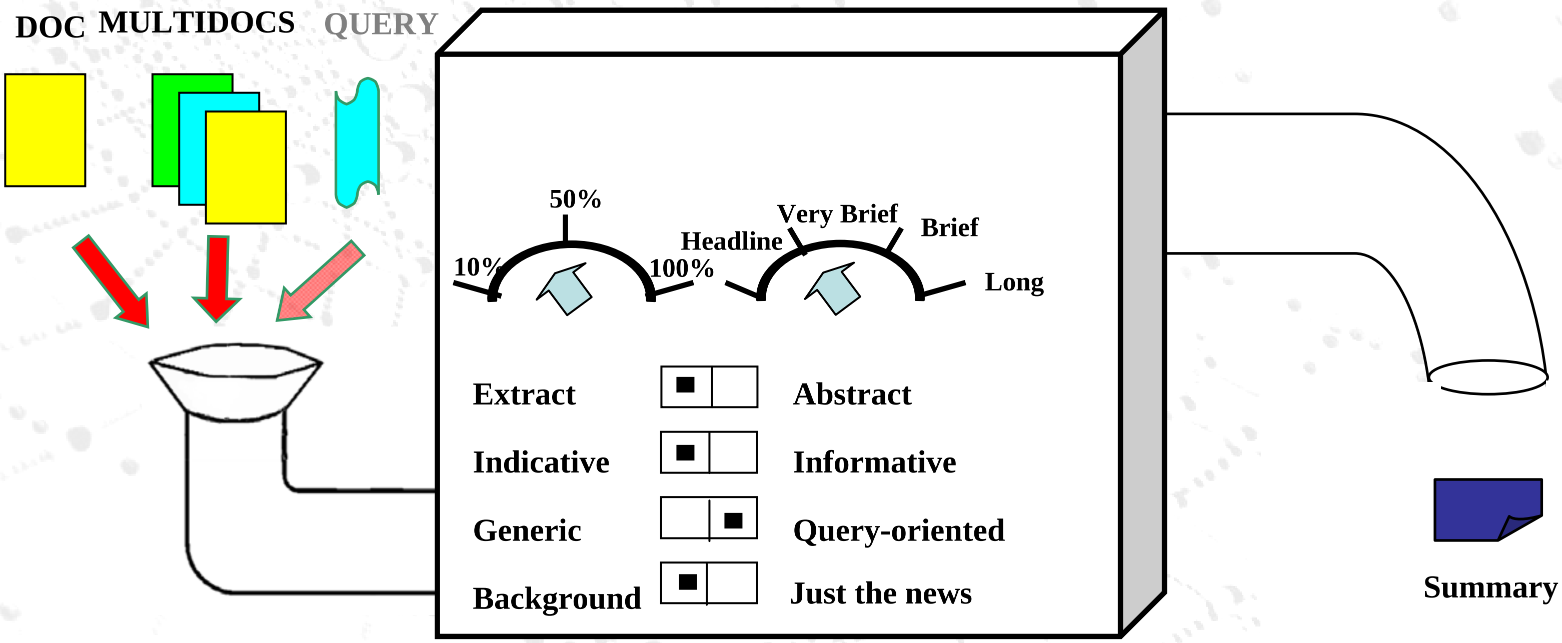
- Question Answering as exact answer retrieval
  - Different types of QA: factoid, list, definitional
- Less volume of information allows more intensive statistical NLP to be applied
  - Pre-process: question typing
  - Post-process: answer extraction
  - Successive Constraint Relaxation to expand queried to find less exact answers.
- Use structure
  - Associating terms into groups (keep in mind for clustering later)
  - Soft patterns for capturing context in an unsupervised way using PRF
- Definitional QA – really summarization in disguise?

Outline

- Summarization - hints people apply
- Introduction to machine learning

- An unsupervised clustering approach
- PageRank in Summarization (Erkan and Radev)
- Editing methods
  - Aligning summaries to extracts (Jing and McKeown)
  - Sentence Compression (Knight and Marcu)
- Evaluation and results

A Summarization Machine



Generate a summary given a text document

Summarization defined

- Definitions
  - Take a text document, extract content from it and present the most important content to the user in a condensed form and in a manner sensitive to the user's or application's needs
- Three approaches to summarization
  - Heuristic-based, supervised and unsupervised learning
  - Each has its problems and advantages

Simplifying the task

- The general task requires:
  1. understanding the meaning of a text document
  2. generating fluent text summary

- Simplified task: Select important sentences verbatim from the input text to form a summary
  - Input: A text document
  - Output: Top *n* sentences with the highest numeric scores (each sentence in the input document is assigned a numeric score

  - Quick question: is this the result an abstract or extract?

Modeling humans

- Studies of human summarizers
  - Cremmins (65) & Endres-Niggemeyer (98) showed that professional summarizers used a number of clues to pick important sentences.
- What do you think these clues were?

Sentence position

Claim: Important sentences occur at the beginning (and/or end) of texts.

- Lead method: just take first *n* sentences!
- Experiments:
  - In 85% of 200 individual paragraphs the topic sentences occurred in initial position and in 7% in final position (Baxendale 58).
  - Only 13% of the paragraphs of contemporary writers start with topic sentences (Donlan 80).

Title words

Claim: Words in titles and headings are positively relevant to summarization.

- Shown to be statistically valid at 99% level of significance (Edmundson 68).
- Empirically shown to be useful in summarization systems.

Cue phrases

Claim: Important sentences contain “bonus phrases”, such as significantly, *In this paper we show*, and *In conclusion*, while non-important sentences contain “stigma phrases” such as *hardly* and *impossible*.

- Method: Add to sentence score if it contains a bonus phrase, penalize if it contains a stigma phrase.

Word frequency

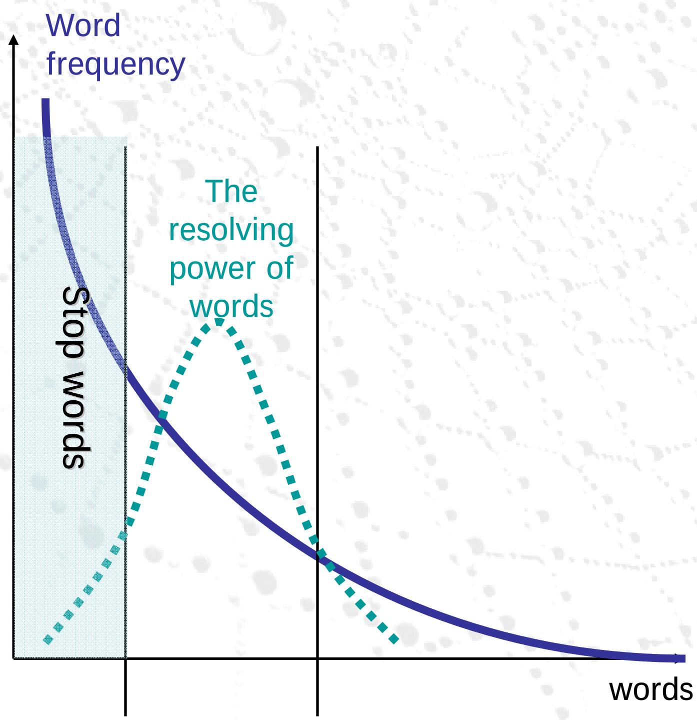


Figure from (Luhn 59)

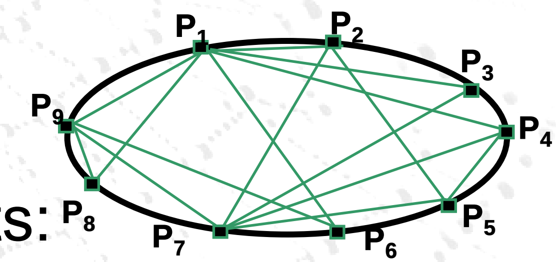
- Claim: Important sentences contain words that occur “somewhat” frequently.
- Method: Increase sentence score for each frequent word.

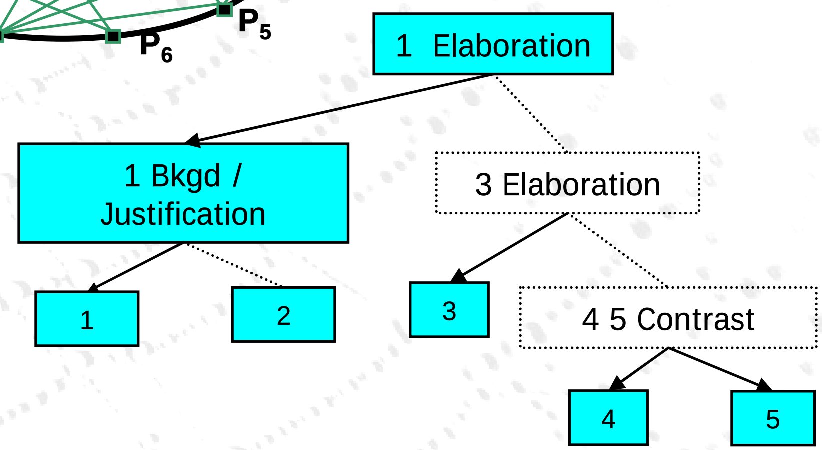
Sentence length

- Claim: both usually long and short sentences aren't usually good for summaries.
  - Long: too much detail, confusing sentence structure or transcribed speech
  - Short: likely to be a section header
- Method: penalize if sentence too long or short.

Discourse hints

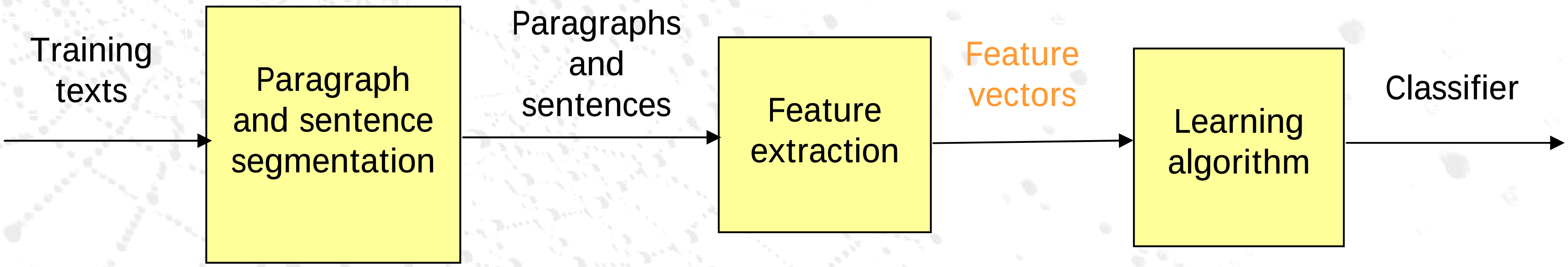
- Claim: flow of topics reflected by the vocabulary and syntactical constructions used.
  - Word overlap: (Mitra *et al*. 97)
  - Discourse and chaining of concepts: (Marcu 97)



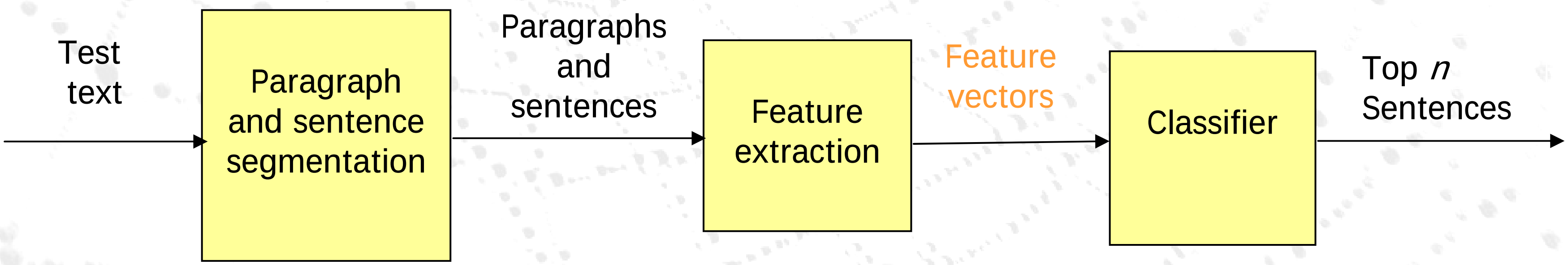


Architecture

- Training



- Testing



Machine learning in 45 minutes or less

Inductive learning

Simplest form: learn a function from examples

- *f* is the target function
- An example is a pair (*x*, *f(x)*)

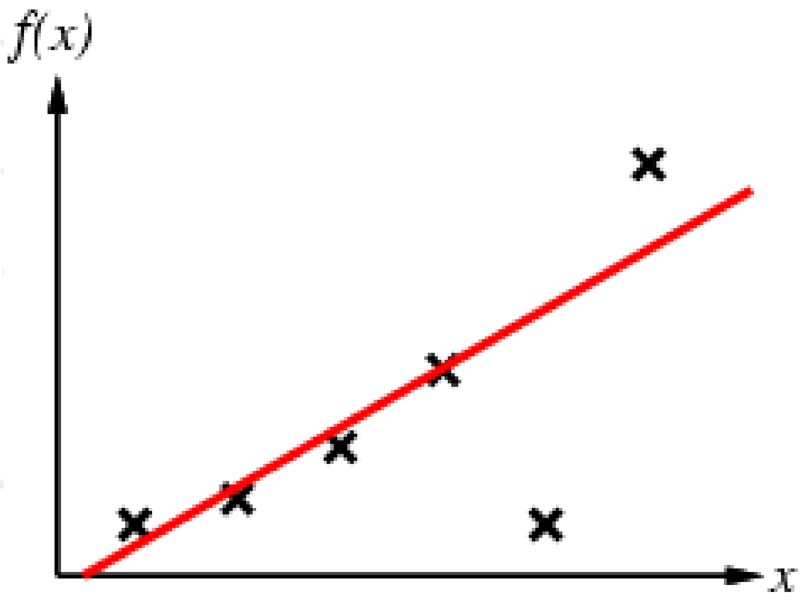
- Problem: find a hypothesis *h*
  such that *h* ≈ *f*
  given a training set of examples

- Many learners do this by constructing a generalized representation of the training set called a model

Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)

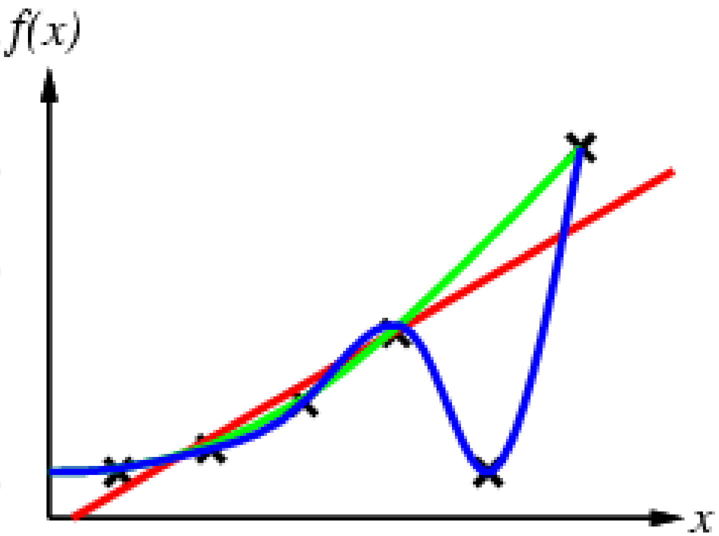
- E.g., curve fitting:



Inductive learning method

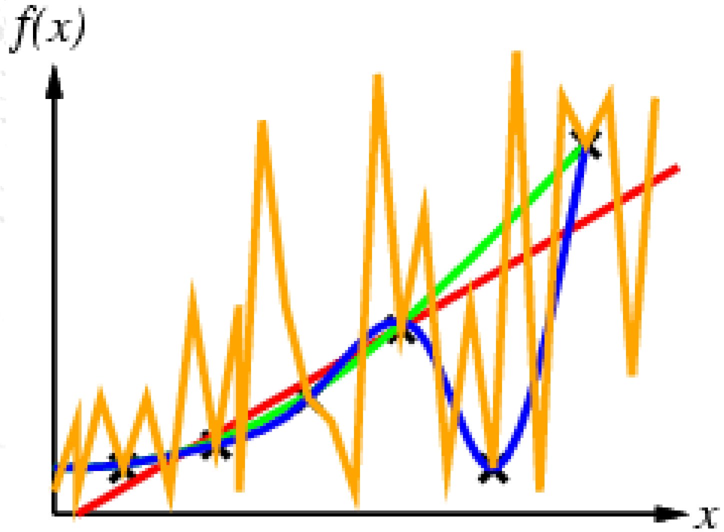
- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)

- E.g., curve fitting:



Inductive learning method

- What's to stop us from predicting this?



- Ockham's razor: prefer the simplest hypothesis consistent with data

Turn your task into a learning problem

Many tasks can be transformed into a learning problem

- Transform the data into features
- Represent the outcomes as a classification task

Overview of learning

- Learners deal with multiple pieces of evidence
  - x can be a vector of values instead of a single value
  - These vectors can be very large
  - Length of the vector = dimensionality
- Learners deal with numeric data
  - Textual data has to be transformed into numeric features
  - Each text token can be reflected as a separate vector
- Learners deal with a fixed set of classes
  - (e.g., f(x) = {finance, politics, sports}
  - But some do this by decomposing multiple classes into n way binary problems, not always optimal

Procedure

Annotation (tedious part)

- Determine data set and classification
- Label the data with the correct classifications
  - This can sometimes be done semi-automatically

Coding (thinking part)

- Code features related to the classification
- Choose an appropriate learning algorithm

Test time

- Split datasets into training and testing portions
- Determine training and testing error
- Analyze errors

Training and testing sets

- Where does the test set come from?

1. Collect a large set of examples
2. Divide into **training** and **testing data**
3. Train on training data, assess on testing
4. Repeat 1-3 for different splits of the set.

The above is called **cross-validation**.

- Must be from the same distribution!!

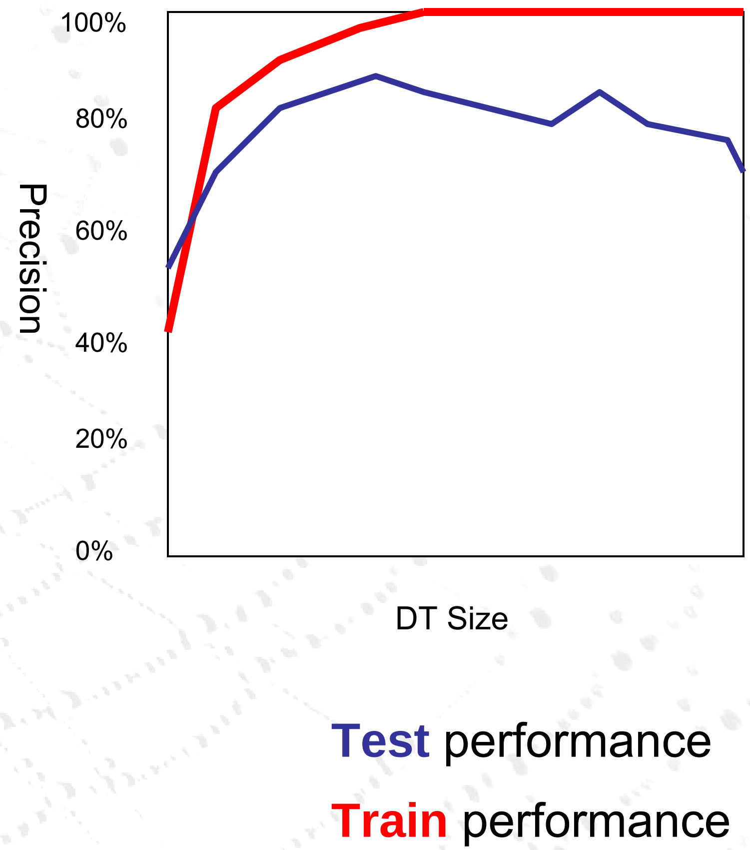
“Learning … enable[s] the system to do the task or tasks drawn from the **same population**” – Herb Simon

– To think about: Why?

– Related area: domain adaptation

Overfitting

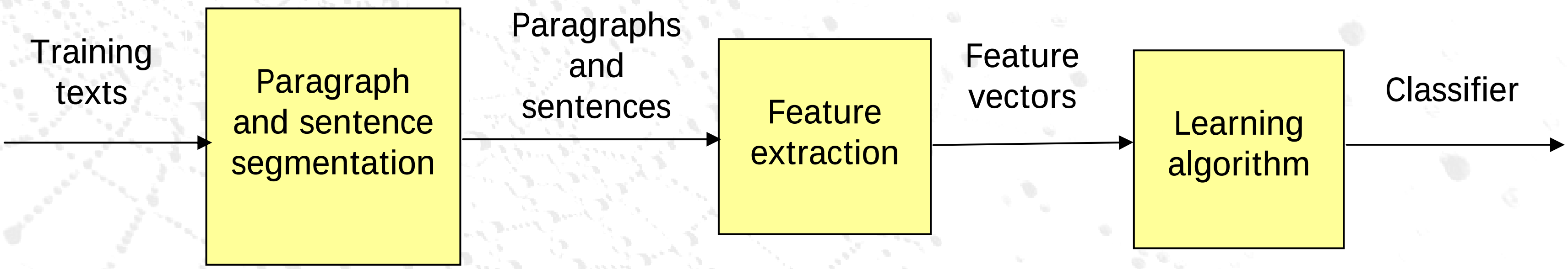
- Better training performance = test performance?
- Nope. Why?
  1. Hypothesis too specific
  2. Models noise
- Pruning
  - Keep complexity of hypothesis low
  - Stop splitting when:
    1. IC below a threshold
    2. Too few data points in node



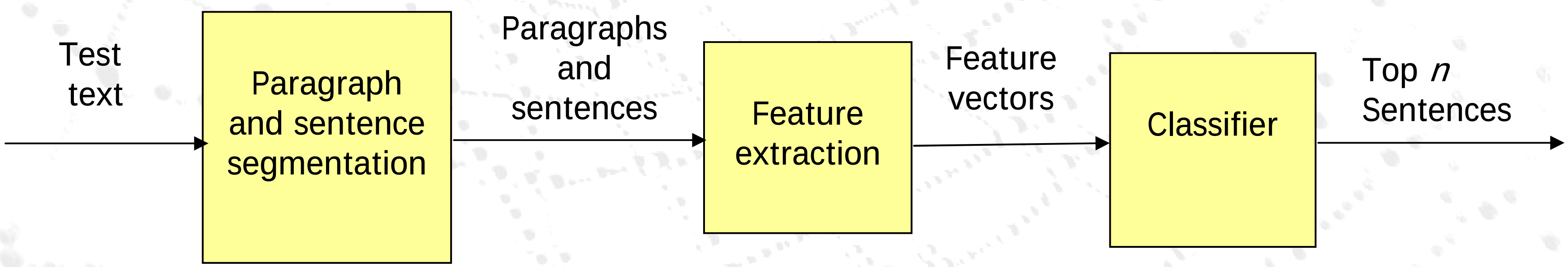
Back to Summarization Methods

Architecture

- Training



- Testing



An unsupervised approach

- Cluster sentences into natural clusters (representing topics) and select a representative sentence from each cluster? (Nomoto & Matsumoto 01)
- Novelty of this approach: Diversity-based Summarization
  - The goal is to find a subset of sentences so as to minimize repetitive concepts (*redundancy*) and to maximize topical coverage (*diversity*)
- General idea:
  1. Find Diversity – Group related sentences into clusters
  2. Reduce Redundancy – For each cluster, identify the most important sentence as a representative

K-means clustering

- Arbitrarily choose *k* initial cluster centroids: $\underline{\mu}_1, \underline{\mu}_2, \ldots, \underline{\mu}_k$

- Repeat until centroid locations converge...

  - Distribute each input vector $\underline{x}$ to the nearest cluster C_i:

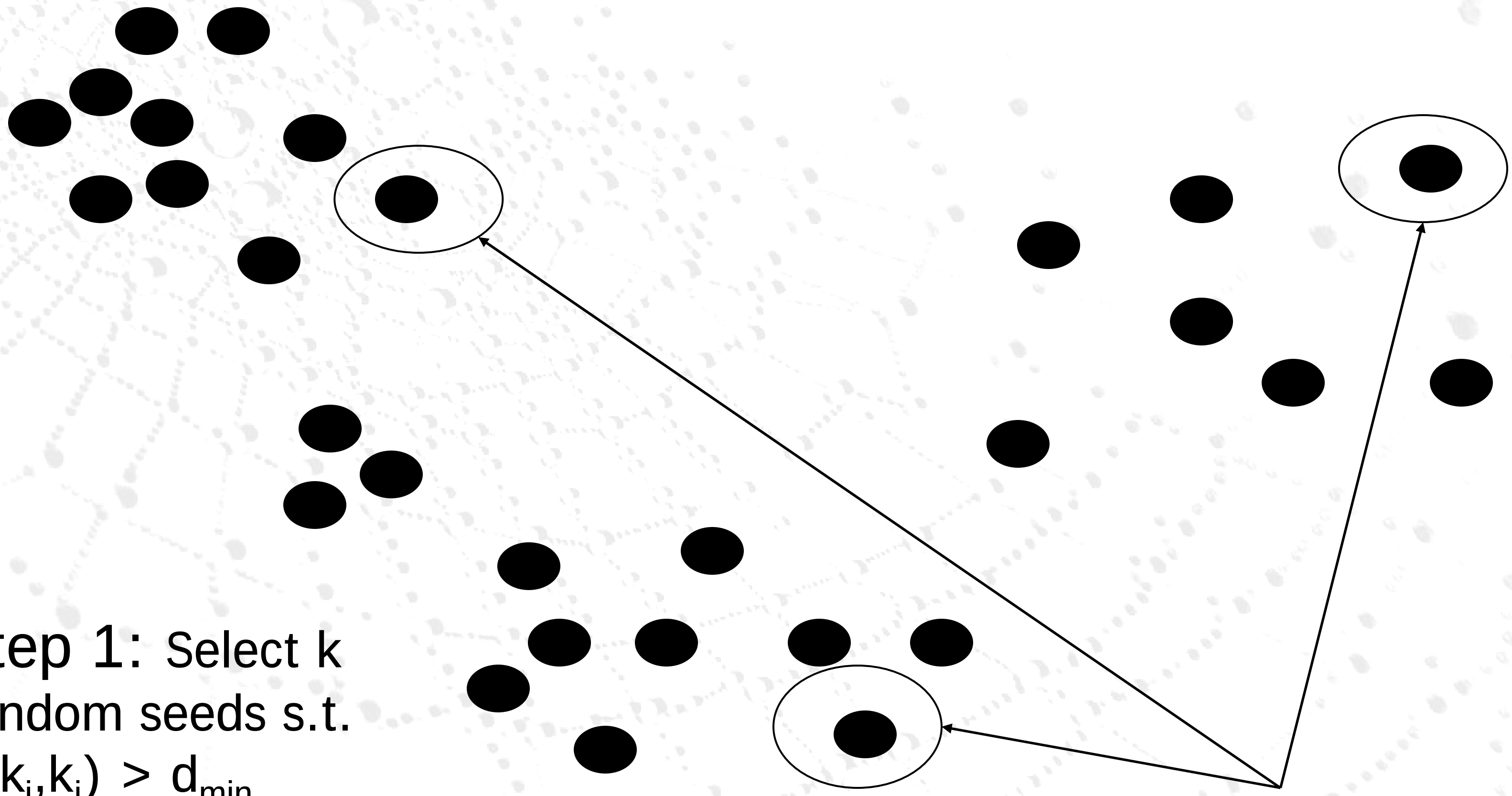
    $\underline{x} \in C_i$ if $d(\underline{x}, \underline{\mu}_i) < d(\underline{x}, \underline{\mu}_j)$ for all $j \neq i$

    where $d(\underline{x}, \underline{\mu}_i)$ is any distance measure

  - Update each cluster centroid:

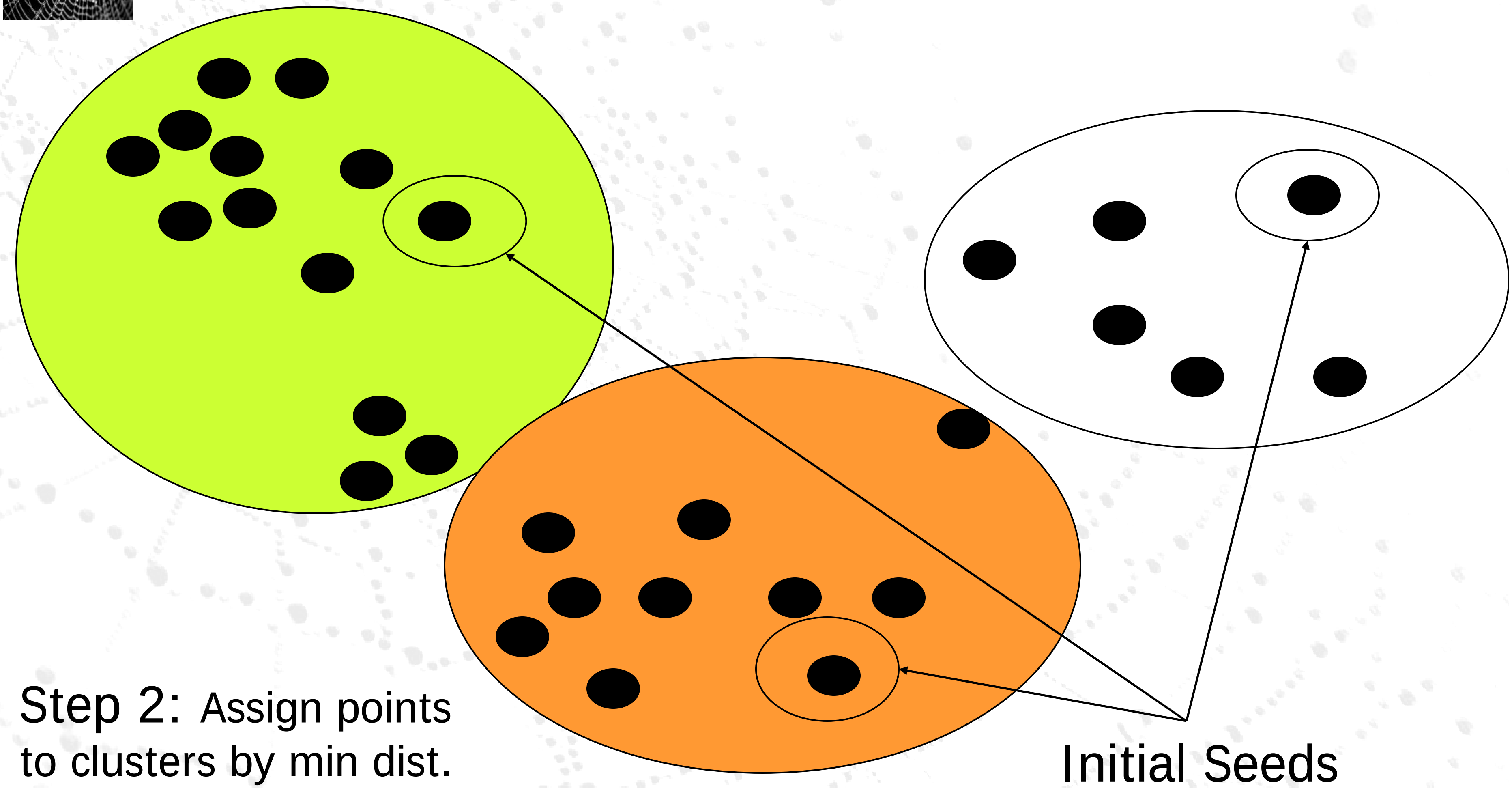
    $$\underline{\mu}_i = \left(\sum_{\underline{x} \in C_i} \underline{x}\right) / |C_i|$$

Initial data points



Step 1: Select k random seeds s.t. $d(k_i,k_j) > d_{min}$

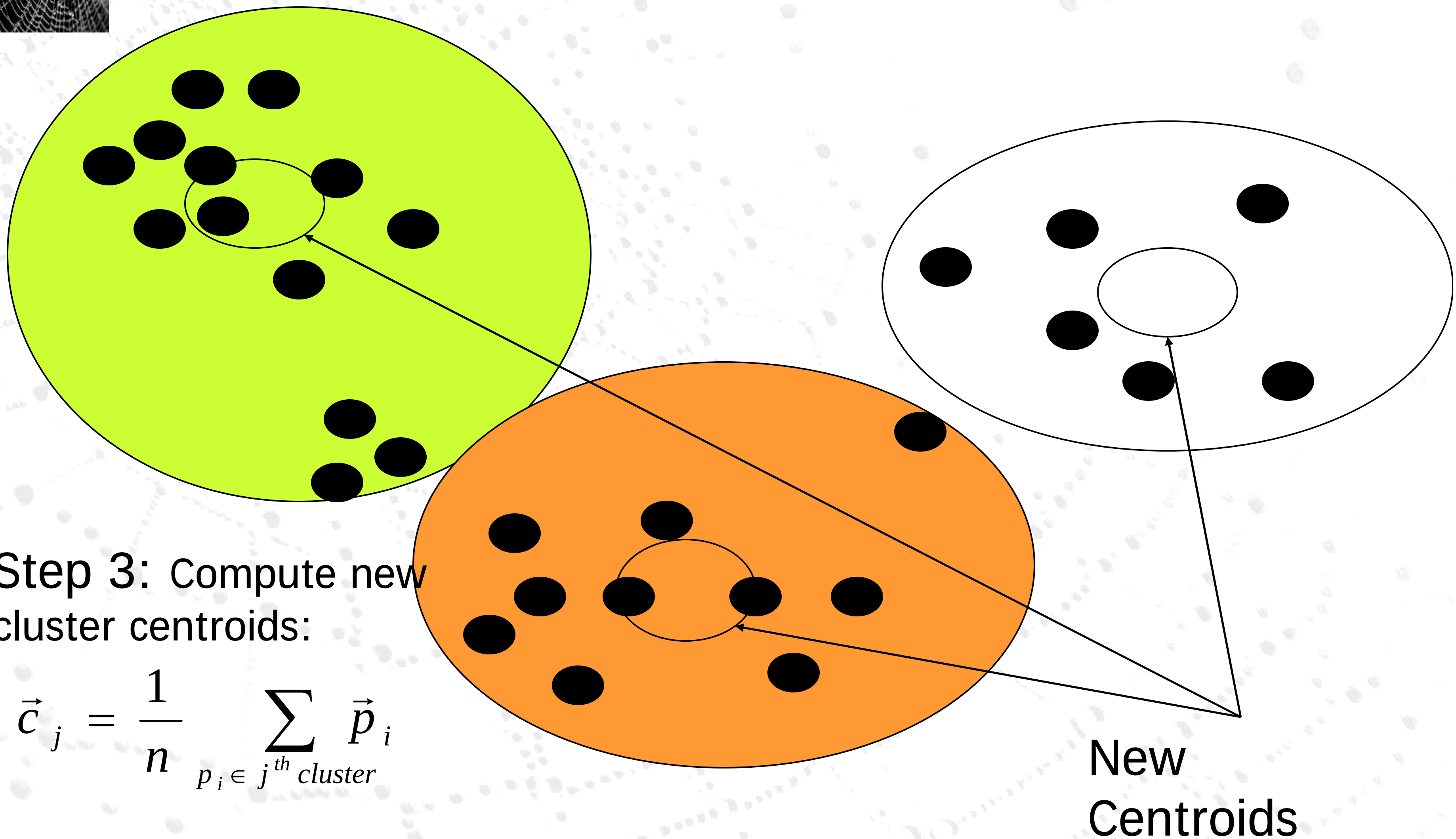
First-pass clusters



Step 2: Assign points to clusters by min dist.

$$\text{Cluster}(p_i) = \underset{s_j \in \{s_1,\ldots,s_k\}}{\text{Argmin}}(d(p_i,s_j))$$

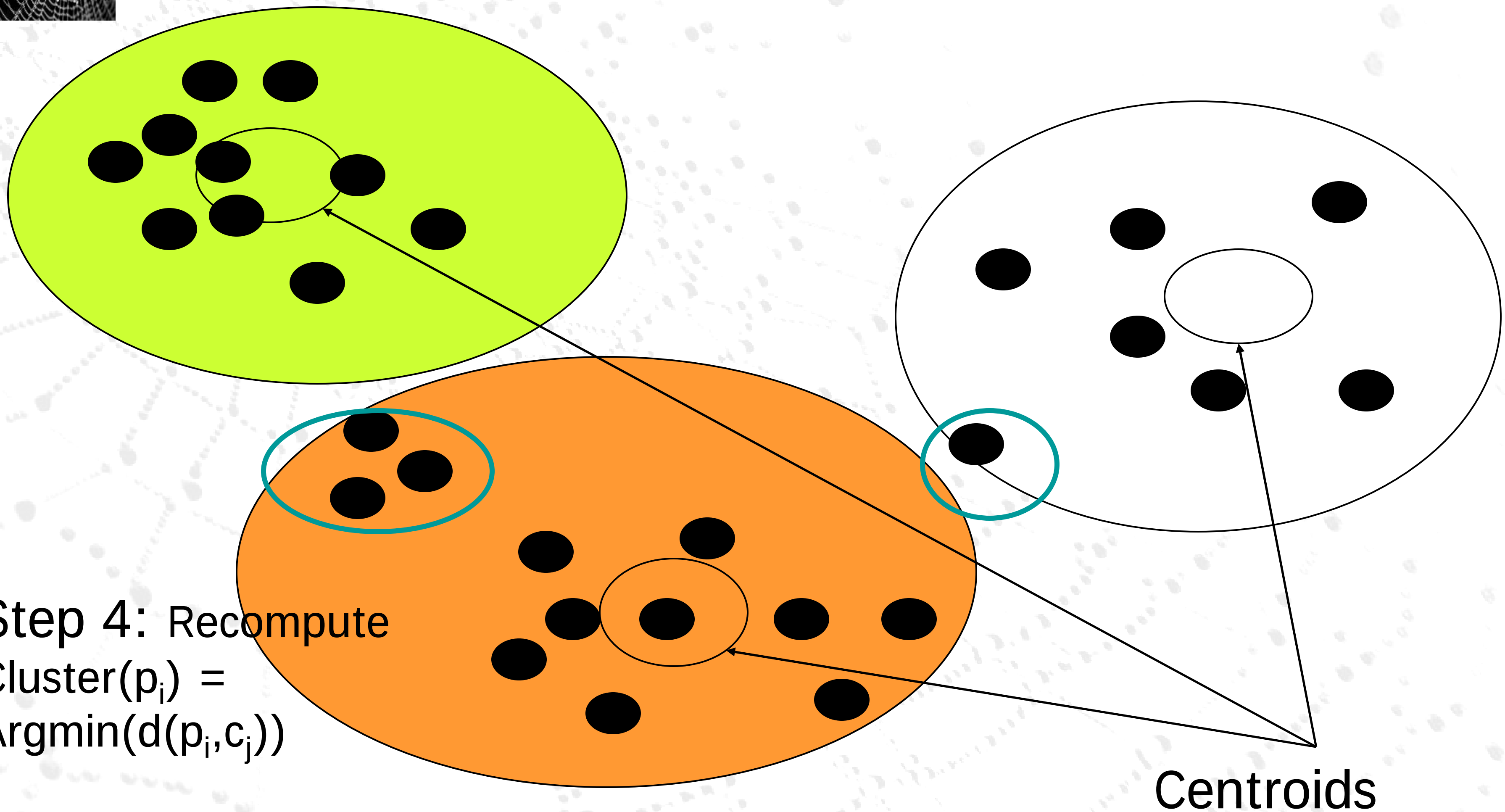
Seeds → Centroids



Step 3: Compute new cluster centroids:

$$\vec{c}_j = \frac{1}{n} \sum_{p_i \in j^{th}\ cluster} \vec{p}_i$$

Second pass clusters

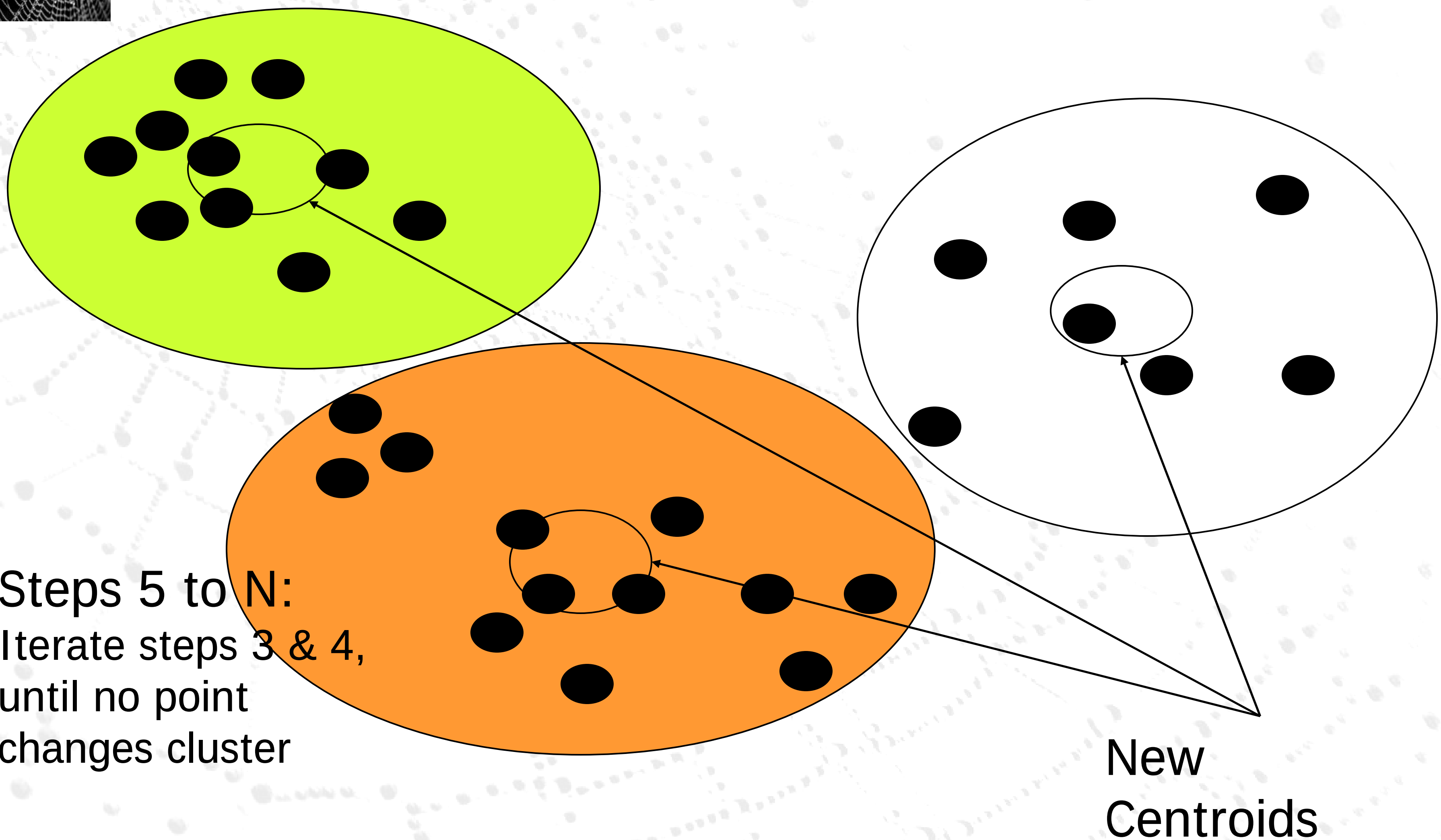


Step 4: Recompute

$\text{Cluster}(p_i) = \text{Argmin}(d(p_i,c_j))$

$c_j \in \{c_1,\ldots,c_k\}$

Iterate until stable



Steps 5 to N:

Iterate steps 3 & 4,
until no point
changes cluster

X-means clustering

- Problems with *K*-means clustering
  1. Need to supply the number of clusters *k*
  2. A bad choice of initial estimates for clusters can have adverse effects on the performance

- Solution to problem #1
  - Start with 2 clusters, repeatedly split each cluster into 2 until maximum # of clusters is reached or the process has converged
- Solution to problem #2
  - Repeatedly run K-means with random initial points and select a solution with minimum *distortion* (a measure of tightness of clusters)

Simplified X^M-means

X^M-means ($\underline{c}_0$, K_{max})

k=2; C = 2-means($\underline{c}_0$)= $\{\underline{c}_1,\underline{c}_2\}$

while $k < K_{max}$ and k does not converge

$S = \{\underline{c} : \underline{c} \in C, \mathbf{F}(_{2\text{-means}}(\underline{c})) < \mathbf{F}(\underline{c})\}$

if S is not empty then

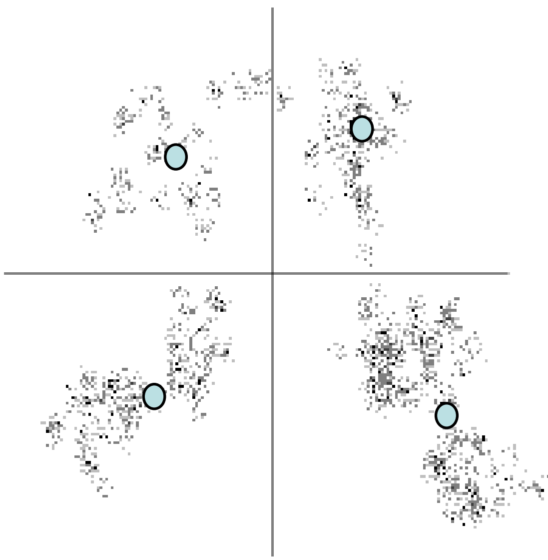
select and split best candidate, $\underline{c}_{best}$, & update cluster:

$C = C \setminus \{\underline{c}_{best}\} \cup {}_{2\text{-means}}(\underline{c}_{best})$

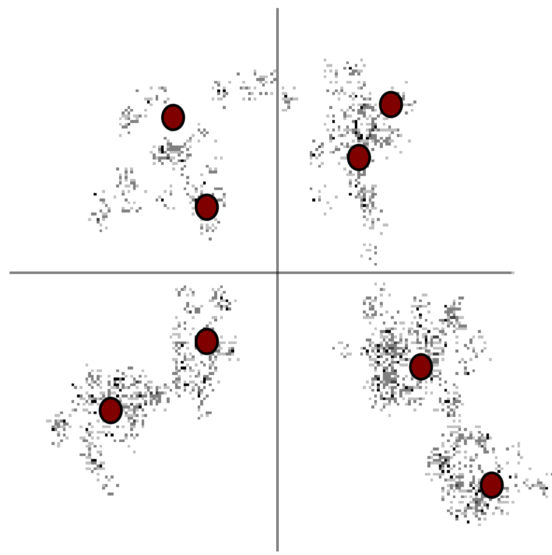
$k = k+1$

- Examples of a function **F** are:
  - Bayesian information criterion (BIC)
  - Minimum description length (MDL)

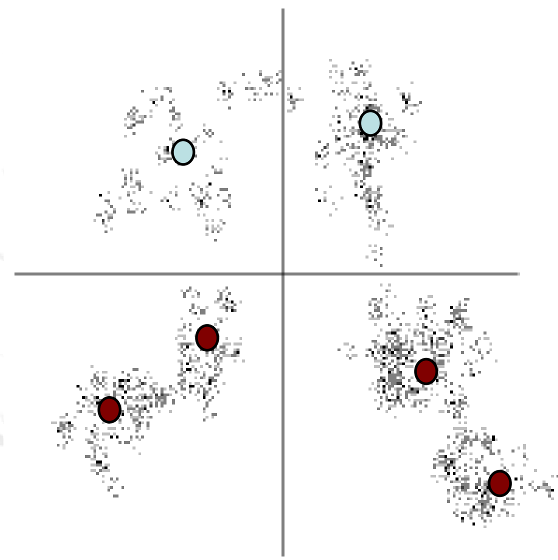
Graphical representation



Initial state with four regions



Each local cluster splits into two sub-regions



Some sub-regions are not worth keeping

Diversity: sentence selection

- After clustering into related sentences, the next task is to pick out the most important sentence per cluster
- For each cluster, assign a weight to each sentence:

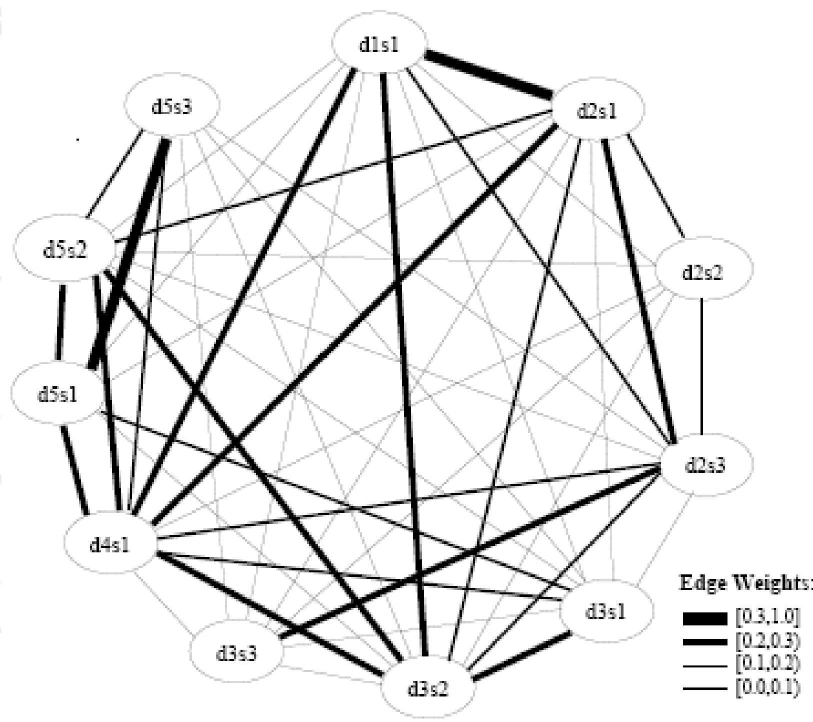
  $\sum_{t \in s} tf \times idf\ (t)$

- Sentences are then ranked in decreasing order of scores
- The best scoring sentence is selected as a representative sentence for that cluster

Graph based summarization

- Idea: Use graphical algorithms for summarization
- View the document(s) as a graph and apply graph algorithms to the set

- Text Unit (e.g., sentence) as a node
- Relationship (e.g., similarity) as an edge
  - Edges weights may be continuous or binary
- Pick out most prominent nodes



PageRank, redux

- Earlier work suggested centrality (by degree i.e., # of edges)
- However, this only models *local* imporance

- Apply PageRank to smooth out importance
- Sort and select top nodes: more needs to be done here
- Two variants used independently by different groups: LexRank (undirected edges + heuristics) and TextRank (directed edges)

Abstracts to extracts?

- Many online technical papers come with an abstract.
  - Can we do automatic alignment to semi-automate the preparation of training material?
- Kupiec's team tried automatic alignment followed by manual human correction
  - Direct match – 1 to 1 correspondence
  - Direct join – 2 to 1 correspondence
  - Incomplete matches – 1 to 2 correspondence
- With the alignment we have extracts for the texts, and both training and evaluation now possible.

Abstractive Summarization

- Abstracts are not necessarily constructed by sentence extraction
  – But an analysis shows that often this is the case (Liddy 1991, Endres-Niggemeyer 2000)
  – Propose cut-and-paste: sentence editing by reduction and combination (Jing 1999)

- Learn this model by aligning abstracts to text in the full paper
  – Then later can apply this to create abstracts.

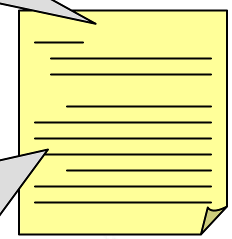
Summary to Text Alignment

Abstract

This	paper	discusses	the

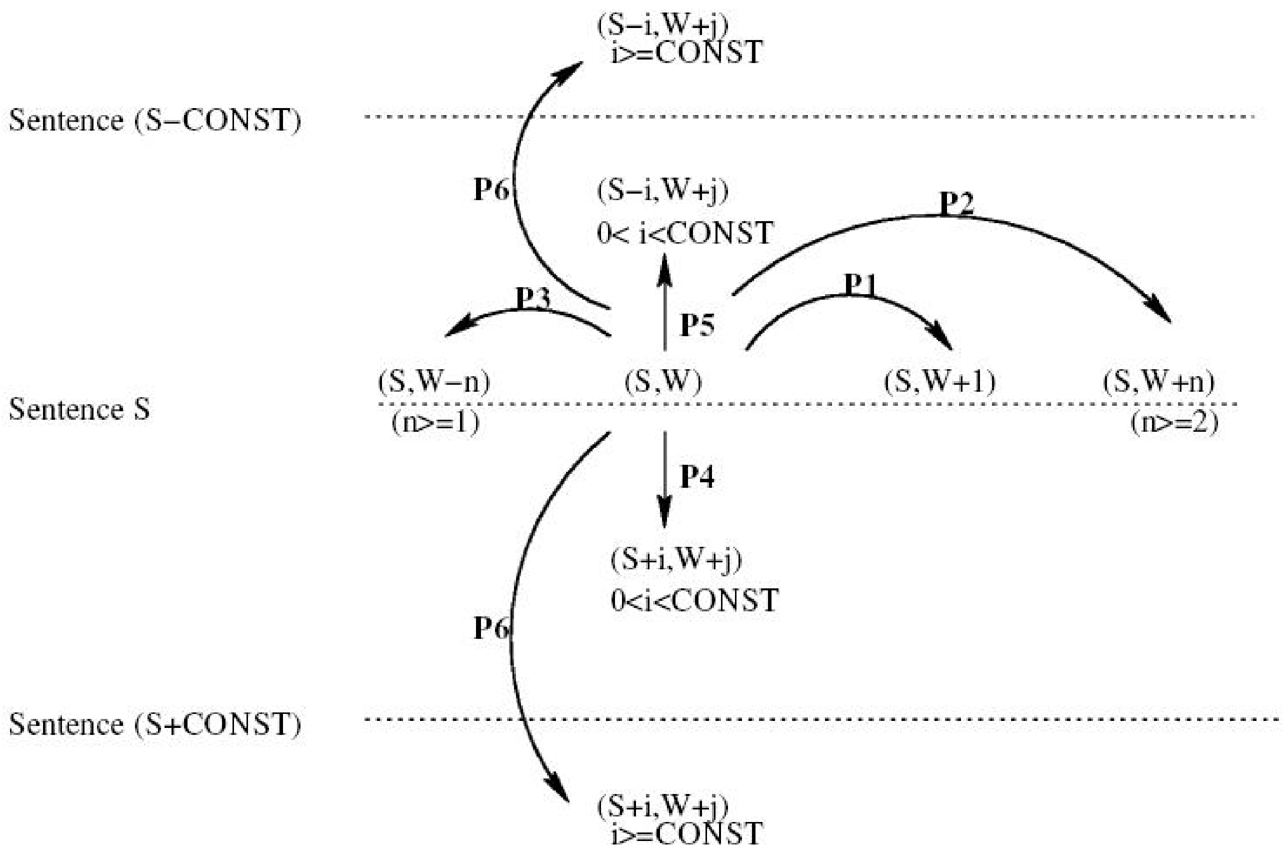
Full text locations

(1,4)	(1,7)	(2, 9)	(2,10)
(1,11)	(1,17)	(2,24)	(2,25)
(2,1)	(2,23)		(3,20)
(2,19)			(4,12)
(2,34)			…



Transition Probabilities

- Assign p1 > p2 > p3 > p4 > p5 > p6



Finding the best path

- Use dynamic programming to find least costly path
  - In Hidden Markov Models (HMM) this is equivalent to the Viterbi algorithm
- This is declared the path that human summarizer used to construct the sentence.

(1,4)	$\xrightarrow{p2}$	(1,7)	$\xrightarrow{p4}$	(2, 9)	$\xrightarrow{p1}$	(2,10)
(1,11)		(1,17)	p1	(2,24)	$\xrightarrow{p1}$	(2,25)
(2,1)	$\xrightarrow{p2}$	(2,23)				(3,20)
(2,19)						(4,12)
(2,34)						…

Bayesian Sentence Compression

- For a given document D, find summary text S that maximizes P(S|D)
- Apply Bayes Rule

$$P(S|D) = P(D|S) \times P(S) / P(D)$$
$$= P(D|S) \times P(S)$$

- How to interpret?
  - – Summary is the source signal
  - – Full document is summary with “noise” added
  - = Noisy channel model

  - – Example: John Doe has secured the votes of most

Noise!

Models for the Noisy Channel

- Source Model to assign P(S): favor grammatically good sentences.
  - Simple method: Use trigram language model
  - Question: why don't we use a unigram or bigram model?
- Channel Model to assign P(D|S): favor D,S pairs where D looks like a plausible expansion of S.
  - Simple method: Favor grammatically plausible additions (e.g., adjuncts) and vocabulary that is optional ("already" → good, "not" → bad)
- Decoder: search through all P(S) and P(D|S) possibilities
  - Non-trivial, as many summaries to consider

Summary evaluation

A first look at evaluation

- An Intrinsic Task:
  - Use number of matching sentences to measure accuracy
  - For each test document, have human generate an *n* sentence summary. The summarizer program also generates an *n* sentence summary.
- Precision = # matching sentences / n gold standard
- Compare against other methods:
  - Use *n* lead sentence as baseline (used by search engines, Microsoft email preview)
  - May also compare with existing summarizer (e.g., MS Word's summarizer)

My summary, your summary

- Human experts tend to have low overlap: about 25% (Rath *et al.* 61)
- What can we do?
  - Ceiling of performance needs to be determined – noisy data
  - Determine which sentences can be considered to convey the same information – assess the utility of adding a sentence to an existing summary

Nomoto & Matsumoto 01

Idea: summary can serve as substitute for source documents in some task

- N&M: evaluate how well summary supports an IR task
  - An example of an extrinsic task

- BMIR-J2 corpus: 5080 news articles in Japanese
  - Articles from diverse domains as economy, engr & industrial technology, but from a single newspaper source
  - 60 queries and associated list of answers
  - Answers of type A (perfect match) and B (relevance to query)
  - Consider both type as relevance in this study

The benefits of diversity

- Summarization systems compared:
  - Baseline – pure tf×idf weighting scheme for all sentences
  - DBS/*K* – diversity using standard K-means clustering
  - DBS/X^M – diversity using X^M-means clustering
  - Full – using original full text

- Results quoted in F-measure for lenient data set (higher is better)

Compression	Baseline	DBS/*K*	DBS/*XM*	FULL
20%	0.095	0.102	0.140	0.170
30%	0.119	0.132	0.146	0.170
40%	0.131	0.143	0.156	0.170
50%	0.147	0.151	0.163	0.170

- Ceiling effect: if bad summaries are good enough for easy tasks... Then we need harder tasks that require better summaries

N-gram summarization evaluation: ROUGE

- Recall-Oriented Understudy for Gisting Evaluation
- Compares quality of a summary by comparison with ideal(s) summaries
- Metrics count the number of overlapping units

ROUGE-N: N-gram co-occurrence statistics is a recall oriented metric

S1- Police killed the gunman
S2- Police kill the gunman
S3- The gunman kill police

S2 equivalent to S3

ROUGE-L: Based on longest common subsequence

S1- Police killed the gunman
S2- Police kill the gunman
S3- The gunman kill police

S2 better than S3

ROUGE experiments

- Co-relation with human judgment
- Experiments on DUC 2000-2003 data
- 17 ROUGE metrics tested
- Pearson's correlation coefficients computed

- Conclusion: ROUGE-1, 2, ROUGE-S, and SU worked well in other multi-doc tasks.

Current trends

- Multi* summarization
  - Multidocument
  - Multilingual
- Revision and regeneration
  - Cut-and-paste summarization
- “Ultra”-summarization for mobile platforms
  - Web surfing on your PDA, handphone
  - Headline, keyword and title generatio

Summary

- Methods
  - Supervised feature based, unsupervised clustering based
  - PageRank for summarization: sentences as nodes, edges as similarity
  - Extractive summarization with smaller units
    - Alignment (Jing) and Bayesian Noisy Channel model
  - Performance around 30-40% as compared to human experts
- Machine Learning in a nutshell
  - Supervised: train and test phrases
  - Learn patterns from vector of features and class
  - Prefer simpler hypotheses
- Evaluation
  - Use n-gram (language models) techniques
  - Other intrinsic methods also, but require more detailed annotation

Basic Summarization References

- **HP Edmunson** (1969). *New methods in automatic abstracting*. J of the ACM 16(2), p 264-285.
- **B Endres-Niggemeyer**. (1998): *Summarizing Information*. Berlin: Springer.
- **G Erkan & D Radev** (2004) *LexRank: Graph-based Lexical Centrality as Salience in Text Summarization*. J. of AI Research. Vol. 22 **H Jing and K McKeown.** (1999) *The Decomposition of Human-Written Summary Sentences.* SIGIR, Berkeley, CA.
- **K Knight and D Marcu** (2000) *Statistics-Based Summarization Step One: Sentence Compression*. Proceedings of AAAI, pages 703-710. **J Kupiec, J Pedersen & F Chen** (1995). *A trainable document summarizer.* ACM SIGIR, 68 – 73. Seattle, WA USA.
- **HP Luhn**. (1958) *The Automatic Creation of Literature Abstracts*. IBM Journal of Research & Development, 2 (2), April, 1958, p 159-165.
- **D Marcu**. (1997) From Discourse Structures to Text Summaries. In ACL/EACL'97 Workshop on Summarization, pages 82-88.
- **M Mitra**, A Singhal & C Buckley (1997) *Automatic text summarization by paragraph extraction*. In ACL'97/EACL'97 Workshop on Summarization, p 39-46.
- **T Nomoto & Y Matsumoto** (2001). *A new approach to unsupervised text summarization.* ACM SIGIR, 26-34