Content Selection: Classification & Graphs

Ling573
Systems & Applications
April 14, 2015

Roadmap

- CLASSY:
  - HMMs
  - Matrix-based selection
  - Linguistic processing
- Graph-based approaches
  - Random walks
- Supervised selection
  - Term ranking with rich features

CLASSY

- "Clustering, Linguistics and Statistics for Summarization Yield"
  - Conroy et al. 2000-2011
- Highlights:
  - High performing system
    - Often rank 1 in DUC/TAC, commonly used comparison
  - Topic signature-type system (LLR)
  - HMM-based content selection
  - Redundancy handling

Using LLR for Weighting

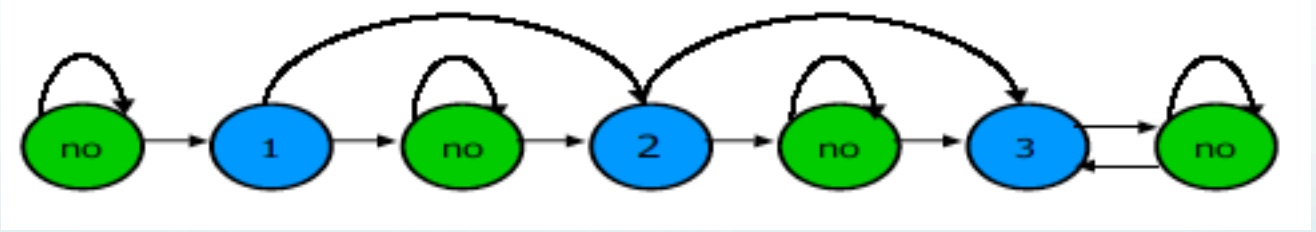
- Compute weight for all cluster terms
  - weight(w_i) = 1 if -2log λ > 10, 0 o.w.
- Use that to compute sentence weights

$$weight(s_i) = \sum_{w \in s_i} \frac{weight(w)}{|\{w|w \in s_i\}|}$$

- How do we use the weights?
  - One option: directly rank sentences for extraction
- LLR-based systems historically perform well
  - Better than tf*idf generally

HMM Sentence Selection

- CLASSY strategy: Use LLR as feature in HMM
- How does HMM map to summarization?
  - Key idea:
    - Two classes of states: summary, non-summary
  - Feature(s)?: log(#sig+1) (tried: length, position,..)
    - Lower cased, white-space tokenized (a-z), stopped
  - Topology:



  - Select sentences with highest posterior (in “summary”)

Matrix-based Selection

- Redundancy minimizing selection
- Create term x sentence matrix
  - If term in sentence, weight is nonzero
- Loop:
  - Select highest scoring sentence
    - Based on Euclidean norm
  - Subtract those components from remaining sentences
  - Until enough sentences
- Effect: selects highly ranked but different sentences
  - Relatively insensitive to weighting schemes

Combining Approaches

- Both HMM and Matrix method select sentences
- Can combine to further improve
- Approach:
  - Use HMM method to compute sentence scores
    - (e.g. rather than just weight based)
      - Incorporates context information, prior states
  - Loop:
    - Select highest scoring sentence
    - Update matrix scores
      - Exclude those with too low matrix scores
    - Until enough sentences are found

Other Linguistic Processing

- Sentence manipulation (before selection):
  - Remove uninteresting phrases based on POS tagging
    - Gerund clauses, restr. rel. appos, attrib, lead adverbs
- Coreference handling (Serif system)
  - Created coref chains initially
  - Replace all mentions with longest mention (# caps)
  - Used only for sentence selection

Outcomes

- HMM, Matrix: both effective, better combined
- Linguistic pre-processing improves
  - Best ROUGE-1,ROUGE-2 in DUC
- Coref handling improves:
  - Best ROUGE-3, ROUGE-4; 2nd ROUGE-2

Graph-Based Models

- LexRank (Erkan & Radev, 2004)

- Key ideas:
  - Graph-based model of sentence saliency
    - Draws ideas from PageRank, HITS, Hubs & Authorities
    - Contrasts with straight term-weighting models
    - Good performance: beats tf*idf centroid

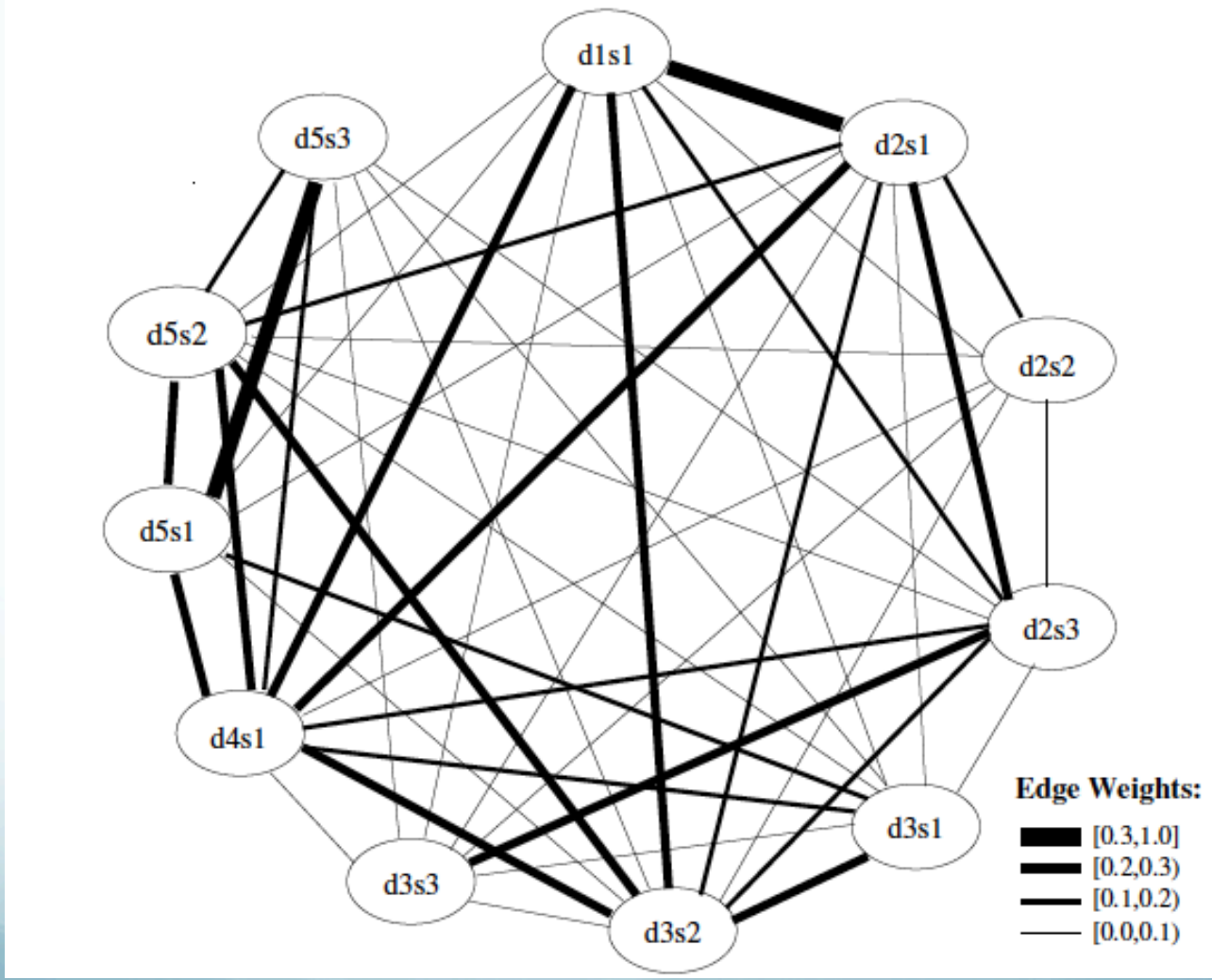
Graph View

- Centroid approach:
  - Central pseudo-document of key words in cluster
- Graph-based approach:
  - Sentences (or other units) in cluster link to each other
  - Salient if similar to many others
    - More central or relevant to the cluster
  - Low similarity with most others, not central

Constructing a Graph

- Graph:
  - Nodes: sentences
  - Edges: measure of similarity between sentences
- How do we compute similarity b/t nodes?
  - Here: tf*idf (could use other schemes)
- How do we compute overall sentence saliency?
  - Degree centrality
  - LexRank

Example Graph



Degree Centrality

- Centrality: # of neighbors in graph
  - Edge(a,b) if cosine_sim(a,b) >= threshold
- Threshold = 0:
  - Fully connected → uninformative
- Threshold = 0.1, 0.2:
  - Some filtering, can be useful
- Threshold >= 0.3:
  - Only two connected pairs in example
  - Also uninformative

LexRank

- Degree centrality: 1 edge, 1 vote
  - Possibly problematic:
    - E.g. erroneous doc in cluster, some sent. may score high
- LexRank idea:
  - Node can have high(er) score via high scoring neighbors
    - Same idea as PageRank, Hubs & Authorities
      - Page ranked high b/c pointed to by high ranking pages
    -

$$p(u) = \sum_{v \in adj(u)} \frac{p(v)}{\deg(v)}$$

Power Method

- Input:
  - Adjacency matrix M
- Initialize p_0 (uniform)
- t=0
- repeat
  - t= t+1
  - $p_t=M^T p_{t-1}$
- Until convergence
- Return p_t

LexRank

- Can think of matrix X as transition matrix of Markov chain
  - i.e. X(i,j) is probability of transition from state i to j
- Will converge to a stationary distribution (r)
  - Given certain properties (aperiodic, irreducible)
  - Probability of ending up in each state via random walk
- Can compute iteratively to convergence via:

$$p(u) = \frac{d}{N} + (1-d) \sum_{v \in adj(u)} \frac{p(v)}{\deg(v)}$$

  - "Lexical PageRank" ➔ "LexRank
  - (power method computes eigenvector)

LexRank Score Example

- For earlier graph:

ID	LR (0.1)	LR (0.2)	LR (0.3)	Centroid
d1s1	0.6007	0.6944	1.0000	0.7209
d2s1	0.8466	0.7317	1.0000	0.7249
d2s2	0.3491	0.6773	1.0000	0.1356
d2s3	0.7520	0.6550	1.0000	0.5694
d3s1	0.5907	0.4344	1.0000	0.6331
d3s2	0.7993	0.8718	1.0000	0.7972
d3s3	0.3548	0.4993	1.0000	0.3328
d4s1	1.0000	1.0000	1.0000	0.9414
d5s1	0.5921	0.7399	1.0000	0.9580
d5s2	0.6910	0.6967	1.0000	1.0000
d5s3	0.5921	0.4501	1.0000	0.7902

Continuous LexRank

- Basic LexRank ignores similarity scores
  - Except for initial thresholding of adjacency
- Could just use weights directly (rather than degree)

$$p(u) = \frac{d}{N} + (1-d) \sum_{v \in adj(u)} \frac{\cos sim(u,v)}{\sum_{z \in adj(v)} \cos sim(z,v)} p(v)$$

Advantages vs Centroid

- Captures information subsumption
  - Highly ranked sentences have greatest overlap w/adj
  - Will promote those sentences
- Reduces impact of spurious high-IDF terms
  - Rare terms get very high weight (reduce TF)
  - Lead to selection of sentences w/high IDF terms
  - Effect minimized in LexRank

Example Results

- Beat official DUC 2004 entrants:
  - All versions beat baselines and centroid
  - Continuous LR > LR > degree
    - Variability across systems/tasks

	2004 Task2		
	min	max	average
Centroid	0.3580	0.3767	0.3670
Degree (t=0.1)	0.3590	0.3830	0.3707
LexRank (t=0.1)	0.3646	0.3808	0.3736
Cont. LexRank	0.3617	0.3826	0.3758

baselines: random: 0.3238
lead-based: 0.3686

(b)

  - Common baseline and component