# Automatic Summarization

## CS 769 Guest Lecture

Andrew B. Goldberg
goldberg@cs.wisc.edu

Department of Computer Sciences
University of Wisconsin, Madison

February 22, 2008

# Outline

- Introduction to summarization
- Keyphrase extraction and document summarization
- Supervised vs unsupervised techniques
- Recent local work: GRASSHOPPER

# Introduction

Automatic summarization: reducing text document(s) into a small set of words or sentences that convey the central meaning of the text.

## Example text

The Atlantis shuttle has undocked from the International Space Station in preparation for its return to Earth. The orbiter detached from the platform at 0924 GMT, ending eight days of operations at the ISS. The crew will perform a final check on the ship's heat-shield system before attempting a landing on Wednesday.

# Introduction

Automatic summarization: reducing text document(s) into a small set of words or sentences that convey the central meaning of the text.

### Example text

The Atlantis shuttle has undocked from the International Space Station in preparation for its return to Earth. The orbiter detached from the platform at 0924 GMT, ending eight days of operations at the ISS. The crew will perform a final check on the ship's heat-shield system before attempting a landing on Wednesday.

### Example summary

Atlantis shuttle has undocked and is preparing for journey home.

# Introduction

General forms of summarization:

- Extractive: select subset of words/phrases/sentences as summary
- Abstractive: parse text and generate natural-sounding summary

Particular types of summarization:

- Keyword extraction:
  produce single words or short phrases to “tag” a document
- Document summarization:
  produce short paragraph summary of a long document

Other dimensions: supervised or unsupervised

# Keyword Extraction

- Given: pieces of text (e.g., news article)
- Do: produce list of keywords/phrases that capture main topics
- Why: help document browsing, better indexing for IR, etc.

# Keywords Example

### Example text

The Army Corps of Engineers, rushing to meet President Bush's promise to protect New Orleans by the start of the 2006 hurricane season, installed defective flood-control pumps last year despite warnings from its own expert that the equipment would fail during a storm, according to documents obtained by The Associated Press.

# Keywords Example

## Example text

The Army Corps of Engineers, rushing to meet President Bush's promise to protect New Orleans by the start of the 2006 hurricane season, installed defective flood-control pumps last year despite warnings from its own expert that the equipment would fail during a storm, according to documents obtained by The Associated Press.

## Extractive keywords

"President Bush," "New Orleans," "defective flood-control pumps"

# Keywords Example

## Example text

The Army Corps of Engineers, rushing to meet President Bush's promise to protect New Orleans by the start of the 2006 hurricane season, installed defective flood-control pumps last year despite warnings from its own expert that the equipment would fail during a storm, according to documents obtained by The Associated Press.

## Extractive keywords

"President Bush," "New Orleans," "defective flood-control pumps"

## Abstractive keywords

"political negligence", "inadequate flood protection"

# Evaluation of Keyword Extractors

Evaluation usually involves precision and recall

- Precision: fraction of proposed keyphrases are actually correct
- Recall: fraction of true keyphrases your system proposed
- F-score is their harmonic mean: $F = \frac{2PR}{P+R}$
- (Note: precision and recall often computed after stemming or applying some other text normalization)

# Keyword Extraction using Supervised Learning

Given: training documents with manually-assigned keywords

- Construct examples out of various text units

# Keyword Extraction using Supervised Learning

Given: training documents with manually-assigned keywords

- Construct examples out of various text units
  - All unigrams, bigrams, trigrams (Turney)
  - Use part-of-speech tags to filter potential examples (Hulth)
  - (Ideally, the examples include the true keywords)
- Compute features:

# Keyword Extraction using Supervised Learning

Given: training documents with manually-assigned keywords

- Construct examples out of various text units
    - All unigrams, bigrams, trigrams (Turney)
    - Use part-of-speech tags to filter potential examples (Hulth)
    - (Ideally, the examples include the true keywords)
- Compute features:
    - Length of text unit
    - Syntactic features of the text unit (all CAPS, contains numbers, etc)
    - Frequency within the current text or larger corpus
    - Relative position of first occurrence in text
- Train a classifier or model (decision trees, naive Bayes, SVM, etc)

# Generating Keywords after Training

Given: test documents

- Construct examples and features exactly as during training
- Apply classifier to each example text unit
- If output generates probabilities or scores, apply threshold
- If it's a binary classifier, just take positive predictions as keywords

Note: Turney's GenEx slightly different:

- Genetic algorithm tunes parameters with training data
- Extractor applies parametrized heuristics to select keywords

# Supervised Keyword Extraction Wrap-up

Pros:

# Supervised Keyword Extraction Wrap-up

Pros:

- may produce interpretable rules
- binary classifier may automatically decide number of keywords

Cons:

# Supervised Keyword Extraction Wrap-up

Pros:

- may produce interpretable rules
- binary classifier may automatically decide number of keywords

Cons:

- need lots of training data
- domain dependent (i.e., biomedical keywords are different than news keywords)

# Keyword Extraction using Unsupervised Learning

Unsupervised methods (i.e., TextRank):

- No training data
- Exploits intrinsic structural properties of text
- Finds keywords that appear “central” (like PageRank)
- Can apply to any text without prior training
- Domain independent

# TextRank

- General-purpose ranking algorithm for NLP
- Build graph over text units as vertices
- Undirected, weighted edges based on similarity
- Intuitively, important vertices connect to other important vertices
- Mathematically, importance given by stationary distribution of random walk on the graph (eigenvector with eigenvalue 1)
- Design choices: how to build graph, how many keywords to produce

# Why TextRank Works

- Why does random walk on co-occurrence graph make sense?
- Frequent words may have many different co-occurring neighbors
- These words act as central hubs
- Closely connected hubs (e.g., “learning” and “classification” through “supervised”) reinforce each other

## Summary

Co-occurrence graph has densely connected regions of terms that both *appear often* and *appear in different contexts*.

# Document Summarization

- Like keyword extraction, tries to find essential parts of text
- Text units are now usually sentences or longer passages
- Extractive summarization much more common
- Abstract summaries are much more difficult to produce

# How Summarizers are Evaluated

- *ROUGE (Recall-Oriented Understudy for Gisting Evaluation)*
- Can download tool at http://haydn.isi.edu/ROUGE/
- Measures how well an automatic summary covers the content in human-generated reference summaries
- Can measure overlap based on $n$-grams for $n = 1, 2, 3, 4$:
  $$\text{ROUGE-1(sys,ref)} = \frac{\text{\# unigrams in ref that appear in sys}}{\text{\# unigrams in ref summary}}$$
- Does not really judge fluency, but merely content overlap
- Similar to BLEU for machine translation (precision-based)

# Supervised Learning for Summarization

- Not much to say—very similar to supervised keyword extraction
- Can learn features of summary-worthy sentences
- Position in document is probably important feature
- One difficulty is that manual summaries should also be extractive in order to create positive training examples
- Labeling effort could be expensive

# Unsupervised Learning for Summarization

Typical unsupervised approach:

- Find “centroid” sentence (mean word vector of all sentences); then rank by similarity to centroid

TextRank and LexRank:

- Graph over sentences
- Edge weights: LexRank uses cosine similarity of TF-IDF vectors, TextRank uses word overlap between sentences
- Can threshold cosine values to create sparse graph
- Combine top ranking sentences to form the summary
- Works based on sentences “recommending” similar sentences

# Issues in Multi-document Summarization

- LexRank used to summarize multiple docs on same topic
- Pool all sentences together as nodes in graph

# Issues in Multi-document Summarization

- LexRank used to summarize multiple docs on same topic
- Pool all sentences together as nodes in graph
- Need to try to eliminate redundant sentences from different docs
- Heuristic: discard lower-ranked sentences that are too similar to higher-ranked ones

# Encouraging Diversity in Ranking: GRASSHOPPER

- Addresses redundancy in multi-doc extractive summarization
- Seeks sentences that are “central” and “diverse”
- Instead of heuristic post-processing, GRASSHOPPER handles diversity during the ranking process
- Developed by Zhu, Goldberg, Van Gael, and Andrzejewski at UW-Madison for NAACL-HLT 07

# GRASSHOPPER Algorithm

- General-purpose graph-based ranking algorithm
- Based on *absorbing Markov chain random walks*
- Absorbing random walk: some states are “absorbing” and act as “black holes” that end walk
- Incorporates prior ranking (i.e., sentence position)

# GRASSHOPPER: Intuitively

- Imagine random walker on graph
- Takes steps randomly according to transition matrix
- Some steps teleport walker to far-away part of graph
- Teleporting is based on prior distribution
- Rank 1: highest stationary probability
- *To get diversity: ranked items turn into absorbing states*
- Ranks 2, 3, . . .: expected visit count before absorption
- States diverse w.r.t. ranked items get most visits
- Result: Hopping behavior between soft clusters in the graph

# GRASSHOPPER: Mathematically

Transition matrix:

$$\tilde{P}_{ij} = \frac{w_{ij}}{\sum_{k=1}^{n} w_{ik}} \tag{1}$$

With teleporting:

$$P = \lambda\tilde{P} + (1-\lambda)\mathbf{1}\mathbf{r}^\top \tag{2}$$

With $\lambda < 1$ and $\mathbf{r}_i > 0$, $P$ has unique stationary distribution:

$$\pi = P^\top \pi \tag{3}$$

First item to be ranked: $g_1 = \mathrm{argmax}_{i=1}^{n}\, \pi_i$

# GRASSHOPPER: Mathematically (2)

After turning ranked items (set $G$) into absorbing states:

$$P = \begin{bmatrix} \mathbf{I}_G & \mathbf{0} \\ R & Q \end{bmatrix} \tag{4}$$

$\mathbf{I}_G$ enforces absorbing property; $R$ and $Q$ relate to unranked items. The *fundamental matrix*

$$N = (\mathbf{I} - Q)^{-1} \tag{5}$$

is used to obtain the expected number of visits before absorption:

$$\mathbf{v} = \frac{N^\top \mathbf{1}}{n - |G|} \tag{6}$$

Next item to rank:

$$g_{|G|+1} = \operatorname{argmax}_{i=|G|+1}^{n} v_i \tag{7}$$

# GRASSHOPPER: Comments

Computational costs:

- Expensive matrix inverse can be avoided using Sherman-Morrison-Woodbury formula
- This involves computing the full inverse only once
- Subsequent steps involve small inverses and simple vector algebra

Summary

- Comparable performance to LexRank and other systems using heuristics for diversity
- Unlike other systems, GRASSHOPPER cleanly incorporates a prior ranking (sentence position, which is a strong baseline)

# Conclusion

- Introduced keyword generation and document summarization
- Discussed pros/cons of supervised and unsupervised methods
- Presented recent work dealing with problem of redundancy

Thanks! Any questions?