Automatic Keyword Extraction from Learning Objects

Kino Coursey and Rada Mihalcea

Our Experiments so Far...

- TextRank
  - Graph-based keyword extraction
- Wikifier
  - Algorithm based on the Wikipedia repository
- Combining the two methods
  - Intersection based on the “least common substring”
- All the evaluations carried out on the History course data from Phase I
TextRank: Random walk algorithms for natural language processing


Random Walk Algorithms

- Usually applied on directed graphs
  - From a given vertex, the walker selects at random one of the out-edges
- Given $G = (V, E)$ a directed graph with vertices $V$ and edges $E$
  - $\text{In}(V_i) =$ predecessors of $V_i$
  - $\text{Out}(V_i) =$ successors of $V_i$

$$S(V_i) = (1 - d) + d \sum_{j \in \text{In}(V_i)} \frac{1}{|\text{Out}(V_j)|} S(V_j)$$

$d$ – damping factor $\in [0, 1]$ (usually 0.85)
TextRank for Keyword Extraction

- Store words in vertices
- Use co-occurrence to draw edges
- Rank graph vertices across the entire text
- Pick top N as keywords

An Example

Compatibility of systems of linear constraints over the set of natural numbers
Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and nonstrict inequations are considered. Upper bounds for components of a minimal set of solutions and algorithms of construction of minimal generating sets of solutions for all types of systems are given. These criteria and the corresponding algorithms for constructing a minimal supporting set of solutions can be used in solving all the considered types of systems and systems of mixed types.

Keywords by TextRank: linear constraints, linear diophantine equations, natural numbers, non-strict inequations, strict inequations, upper bounds

Keywords by human annotators: linear constraints, linear diophantine equations, non-strict inequations, set of natural numbers, strict inequations, upper bounds
Previous evaluation on INSPEC abstracts

- Evaluation:
  - 500 INSPEC abstracts
  - collection previously used in keyphrase extraction [Hulth 2003]

- Previous work
  - mostly supervised learning
  - [Hulth 2003]
  - training/development/test : 1000/500/500 abstracts

<table>
<thead>
<tr>
<th>Method</th>
<th>Assigned</th>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Mean</td>
</tr>
<tr>
<td>TextRank</td>
<td>6,784</td>
<td>13.7</td>
</tr>
<tr>
<td>Ngram with tag</td>
<td>7,815</td>
<td>15.6</td>
</tr>
<tr>
<td>NP-chunks with tag</td>
<td>4,788</td>
<td>9.6</td>
</tr>
<tr>
<td>Pattern with tag</td>
<td>7,012</td>
<td>14.0</td>
</tr>
</tbody>
</table>

Text Wikification

- Finding key terms in documents and link them to relevant encyclopedic information.

Wikiification Pipeline

Keyword Extraction

- Semi-Controlled vocabulary
  - Wikipedia article titles and anchor texts (surface forms).
    - E.g. “USA”, “U.S.” = “United States of America”
  - 1,918,830 terms/phrases
  - Vocabulary is broad: “the” has 9 senses.

- Unsupervised keyword extraction
  - Tf * Idf
    - Wikipedia articles as document collection
  - Chi-squared independence of phrase and text
    - The degree to which it appeared more times than expected by chance
  - Keyphraseness:
    \[ P(\text{keyword} \mid W) = \frac{\text{count}(D_{\text{key}})}{\text{count}(D_w)} \]
Previous Evaluation on Wikipedia

- 85 documents containing 7,286 links
- Extract $n$ keywords, $n=6\%$ of number of words

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tf * I df</td>
<td>41.91%</td>
<td>43.73%</td>
<td>42.82%</td>
</tr>
<tr>
<td>Chi-squared</td>
<td>41.44%</td>
<td>43.17%</td>
<td>42.30%</td>
</tr>
<tr>
<td>Keyphraseness</td>
<td>53.37%</td>
<td>55.90%</td>
<td>54.63%</td>
</tr>
</tbody>
</table>

Combining TextRank and Wikify!

- Using the strengths of both systems
  - TextRank focuses on estimating the attention given to terms in the text
  - Wikify focuses on keywords identified by a large number of people (Wikipedia)
  - Each gets a different set of interesting terms
The Violent Agreement Problem

- Two extractors with possibly different but complete segmentations of the same text.
  - “Mexican traveler” vs “Mexican,” “traveler”
  - “Birth of Venus Sandros Botticelli” vs “The Birth of Venus”, “Sandros Botticelli”
- TextRank gets extended noun-chunks while Wikify! gets common key phrases or object identifiers
- Need a principled way to find agreement
  - Intersection, Union, longest common substring

LCS : Longest Common Substring

- Problem: Given sequences \( x[1..m] \) and \( y[1..n] \), find a longest common subsequence of both.
- Example: \( x = BDABCBADAB \) and \( y = BDBCBABDAB \),
  - BCB is a common substring and
  - BCBA and BDAB are two LCSs
- Common problem for aligning two DNA sequences
- Uses a dynamic programming method to find the longest common path through both strings
- In Subsequence (vs Substring) one allows gaps and is related to minimum Edit Distance

http://en.wikibooks.org/wiki/Algorithm_implementation/Strings/Longest_common_substring
http://en.wikipedia.org/wiki/Longest_common_substring_problem
LCS Example

- Applies to “Birth of Venus Sandros Botticelli” vs “The Birth of Venus”, “Sandros Botticelli”
  - LCS(“Birth of Venus Sandros Botticelli”, “Sandros Botticelli”) = “Sandros Botticelli”

- Do a cross comparison for the output of both keyword sources keeping the longest match found for each
- Captures coherent fragments found by both

LCS Algorithm

```
function LCSsubr(S[1..m], T[1..n])
L := array(0..m, 0..n)
z := 0  (length of longest match)
ret := {}  (set of longest matches)
for i := 1..m
  for j := 1..n
    if S[i] = T[j] then L[i,j] := L[i-1,j-1] + 1  (the upper left diagonal)
    if L[i,j] > z then z := L[i,j]   ret := {}     (new longest found)
    if L[i,j] = z then ret := ret ∪ {S[i-z+1..i]} (an equal longest found)
return ret
```

Returns set of all matches of maximal length in one pass through the two strings
Intersection and Union

- Intersection
  - Create a list of words and phrases common to both list
- Union
  - Create a list of words and phrases on either list

The System

Source Text --- Gold Standard Subject List

- Wikifier
  - WK-Articles
  - Wiki-keywords
- TextRank
  - TR-keywords
- Evaluator
  - LCS-keywords
  - Intersection-keywords
  - Union-keywords
  - Statistics

LCS
Intersection
Union
Evaluations on History LO

- Goal: Given the export of the text of the Learning Objects determine the performance of the various methods
  - Precision and recall for basic whole keyword extraction
  - Individual words in the text being correctly classified as being in the bag-of-gold keywords

Gold Standard and its use

- Each collection of learning objects in a directory has a Dublin Core description file with subjects specified. These subjects are the gold standard.
  - Issue 1: One set of keyword for a set of files
  - Issue 2: The set of keywords may not have any direct reference in any of the text
- Each file assumes that the gold for it is the gold found in the appropriate Dublin file
- A pseudo-document is created consisting of all the text in the group to test the Dublin keywords against all the text the Dublin file covers
The Flash Card Problem

- Several html files are labeled “flash cards” and contain the following text: “the flash cards”
- Each card has different gold standard sets
- Each contains the same data
- “the flash cards” is not in the gold standard
- Same data + different specified outcome + no valid clues = ???

- http://lit.csci.unt.edu/~rada/Viewer/the%20flash%20card%205.htm

Relative Recall

Relative recall = number of keywords identified out of gold standard keywords that appear in the text
Keyphrase Precision

keyPrecision1 = Gold keywords found in KeyList / Total Keylist size
keyPrecision2 = KeyList words found in Gold / Total KeyList size

Word Level Precision

Word level precision = True positive / ( True positive + false positive )
= # correct guesses / # guesses
**Word Level Recall**

Recall = True Positive / (True Positive + False negative) = # correct guesses / # gold words

**Word Level F-measure**

F-measure = (2 * True positive) / (2 * true positive + false positive + false negative) = balance between recall and precision
Word Level Accuracy

Accuracy = (True positive + true negatives)/(all words)
= percentage of total word classifications correct

Word Level Error Rate

Error = 1 - Accuracy
= percentage of total possible misclassified
Learning Object Analysis

- Complete Analysis
  - http://lit.csci.unt.edu/~rada/Viewer
- Final Statistics
  - http://lit.csci.unt.edu/~rada/Viewer/SystemFinalSummary.html
- The Flash Card Problem
  - http://lit.csci.unt.edu/~rada/Viewer/the%20flash%20card%208.htm.txt.html
- Boston Tea Party
- Problems Facing the New Country

Thoughts on Performance

- If you want high recall: Wikifier
  - Relative recall: 69%
  - Low precision: 4%
- If you are interested in balance: LCS
  - Recall: 21%
  - Precision: 16%
Questions, Thoughts …

- Gold standard is not really “gold”
- Should we run a separate evaluation with users?
- What will be the end use of the automatic KE?
  - Emphasis on recall vs. emphasis on precision
- ???

Next Step …

- Explore using the WikiArticles and performing WikiRank on them to get related articles and their keywords
  - A way to find higher level concepts not mentioned like “Revolutionary War” or “United States History” from a set of battles.