A Visual Approach to Automated Text Mining and Knowledge Discovery

Doctoral Dissertation
by
Andrey A. Puretskiy
Advisor: Dr. Michael W. Berry
Department of Electrical Engineering and Computer Science
University of Tennessee, Knoxville

November 5, 2010

Motivations

- Vast Quantities of Text Available
  - Scientific Literature
  - News Articles and Blogs
  - Email

- Effective Visual Analytics Requirements:
  - Process Vast Quantities of Textual Information
  - Significant Automation of Analysis
  - Visual, Human-understandable Results Presentation

Dissertation Proposal Revisited

- Integrate visual post-processing and nonnegative tensor factorization (NTF)
- Improve upon existing NTF technique
  - Allow the user to affect factorization by adjusting term weights within the tensor
  - Add automated result classification to visual results post processing
  - Demonstrate effectiveness of approach using several different datasets
- Create an environment for testing of different heuristics for tensor rank estimation

Visual Analytics Environment Architecture
Tensor Factorization

- Tensor: Multidimensional array
- History: Hitchcock (1927), Cattell (1944), Tucker (1966)
- Factorization: Process of rewriting a tensor as a finite sum of lower-rank tensors
- PARAFAC: Parallel Factors Analysis (Harshman, 1970)

Tensor Factorization: PARAFAC Methodology

- Given tensor $X$ and rank $R$, define the factor matrices as combinations of vectors from rank-one components

$$X \approx A \cdot B \cdot C = \sum_{r=1}^{R} a_r \otimes b_r \otimes c_r$$

- Alternating Least Squares:
  Cycle “over all the factor matrices and perform a least-squares update for one factor matrix while holding all the others constant.” (Bader, 2008)

Illustration of a Time-by-Author-by-Term Tensor Decomposition

Nonnegative Tensor Factorization (NTF)

- Nonnegative tensor factorization algorithm: PARAFAC with nonnegativity constraint
- Matlab® Code (Dr. Brett Bader, Sandia)
- Python Translation (Mr. Papa Diaw, Advisor: Dr. Michael Berry)
- Extracts features from textual data
  - Each feature may be described by a list of terms and tagged entities
Performance Comparison

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of Documents</th>
<th>Avg. Document Length (terms)</th>
<th>Matlab NTF Execution Time (minutes)</th>
<th>Python NTF Execution Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kenya 2001-2009</td>
<td>900</td>
<td>696</td>
<td>4.54</td>
<td>17.15</td>
</tr>
<tr>
<td>VAST 2007</td>
<td>1455</td>
<td>391</td>
<td>3.95</td>
<td>16.13</td>
</tr>
</tbody>
</table>

- Times were averaged over 10 trials
- While not as fast as Matlab®, Python still allows real-time analysis
- Future improvements in Python NTF code performance may be possible

NTF: Multidimensional Data Analysis

Build a 3-way array such that there is a term-entity matrix for each time point.

Textual Data (e.g., collection of news articles)

Nonnegative PARAFAC

Third dimension offers more explanatory power: uncovers new latent information and reveals subtle relationships

Sample NTF Output

```
############## Group 15 ##############
Scores      idx  Name
0.2485621   7120 bruce longhorn 7120
0.2485621   7122 longhorn 7122
0.2485621   7128 chainworth 7128
0.2485621   7124 gil 7124
0.2485621   7121 virgina tech 7121
0.2485621   7125 may ann ollesen 7125
...          
Scores      idx  Term
0.298673     6967 monkeypox
0.295470     7468 outbreak
0.2008147     6388 longhorn
0.1594331     4664 gil
0.1592401     1856 chinchilla
0.1434742     11049 travel
0.139183     9322 sars
0.1379675     1857 chinchillas
0.1362139     2372 continent
0.1294388     3888 expect
0.1215461     9711 sick
0.116760     7469 outbreaks
0.1144558     3883 exotic
0.1122035     7232 pets
0.1026513     8088 pot-belled
0.1026513     7229 novelty
0.1019215     1742 cesar
0.1004109     10280 strain
0.1000808     5878 jul
...          
```

FutureLens

**Features**

- Automatically Loads All Terms Found in Input Dataset (except those on the list of exclusions)
- Ability to Search through Terms
- Ability to Sort Terms
- Ability to Create Collections of Terms
- Ability to Create Phrases
Completed Goals

- Integration of Pre-processing, NTF, and FutureLens into a single analysis environment
- Allowing the user to affect the NTF process through Integrated Analysis Environment controls:
  - User is able to define relative importance (or trustworthiness) of terms or subsets of terms
- Introduction of automatic NTF results classification through the use of pre-existing and user-modifiable dictionaries

Integrated Analysis Environment Features and Design Objectives

- Objectives
  - A single application
  - Simple look to avoid feature overload
  - Easy to use without much experience
  - Integration of multiple important capabilities
  - Implemented in Python
    - Portability
      - Linux, OS X, Windows
    - Look and feel of application native to the user’s operating system
    - Easily modifiable due to Python’s excellent readability

Integrated Analysis Environment Capabilities

- Addition of temporal information into the dataset in SGML-tagged format
- User-customized entity tagging (SGML format)
- NTF input file creation
- Tensor term weight adjustment
- Python NTF PARAFAC execution
- FutureLens launching for continuing visual analysis of NTF results

Integrated Analysis Environment
Tensor Term Weights Adjustment

Motivation

- Lack of interest in subset of terms
  - Terms may have been deemed “untrustworthy”
  - Terms may likely be irrelevant to particular analysis model
  - The above may be insufficient to eliminate terms as stopwords
- Strong interest in a subset of terms
  - Subset may have been deemed particularly trustworthy
  - Analyst may need to create a model that focuses strongly on a particular aspect of the data

The Simple Approach

- Plain-text files containing lists of terms
  - Easy for computer-inexperienced users
  - Each file corresponds to a particular analysis model
  - Very easy to create, distribute, view, share feedback, modify models
  - Integrated Analysis Environment quickly creates a term-weight modified NTF input file based on such input

Automated NTF Output Group Labeling

- Motivation: Increase efficiency of human analysis of NTF results
- Automated labeling feature functions much faster than analyst labeling ever could
- Feature allows the analyst to quickly sort NTF output groups by analyst-defined categories
  - Focus exclusively on category or categories of interest
  - Feature includes a default (“none of the above”) category

Automated Labeling Design and Utilization

- Plain-text files containing lists of terms
  - Easy for computer-inexperienced users
  - Very easy to create, distribute, view, share feedback, modify models
- FutureLens quickly labels NTF output groups based on the set of category descriptor files loaded at the time
- Visual category labeling allows the analyst to filter out uninteresting groups and focus on the ones most pertinent to the focus of analysis
Conclusions

- The demonstrated approach can be effectively used to analyze vast quantities of textual data
- The approach is straightforward and easy to use even for computer-inexperienced analysts
- The approach is highly portable and functions under Linux, OS X, and Windows

References

References


Questions?