Textual Influence Modeling Through Non-Negative Tensor Decomposition

Robert Earl Lowe

July 12, 2018
Introduction

- Problem Statement
- Background

Approach

- Model Overview
- Implementation

Results

- A Simple Example
- Analysis of a Conference Paper
Outline

1. Introduction
   - Problem Statement
   - Background

2. Approach
   - Model Overview
   - Implementation

3. Results
   - A Simple Example
   - Analysis of a Conference Paper
Text Documents and Influences

Every text document is a combination of an author’s contributions and contributing factors.
Every text document is a combination of an author’s contributions and contributing factors.

Contributing Factors

Text Documents and Influences
Every text document is a combination of an author’s contributions and contributing factors.

Contributing Factors
- Cited Sources
Text Documents and Influences

- Every text document is a combination of an author’s contributions and contributing factors.
- Contributing Factors
  - Cited Sources
  - Collaborators
Text Documents and Influences

- Every text document is a combination of an author’s contributions and contributing factors.
- Contributing Factors
  - Cited Sources
  - Collaborators
  - Unconscious Influences
Goals and Contributions

- Invent an analysis technique which models:
  
  - Text Document Influencing Factors
  - Text Document Author Contributions
  - Semantics of Influences and Author Contributions

- Create open source software which:
  - Provides efficient handling of large sparse tensors.
  - Allows binding to high level languages.
  - Uses MPI to decompose very large sparse tensors.

Robert Earl Lowe
Textual Influence Modeling
Goals and Contributions

- Invent an analysis technique which models:
  - Text Document Influencing Factors
Goals and Contributions

- Invent an analysis technique which models:
  - Text Document Influencing Factors
  - Text Document Author Contributions
Goals and Contributions

- Invent an analysis technique which models:
  - Text Document Influencing Factors
  - Text Document Author Contributions
  - Semantics of Influences and Author Contributions
Goals and Contributions

- Invent an analysis technique which models:
  - Text Document Influencing Factors
  - Text Document Author Contributions
  - Semantics of Influences and Author Contributions
- Create open source software which:
Invent an analysis technique which models:
- Text Document Influencing Factors
- Text Document Author Contributions
- Semantics of Influences and Author Contributions

Create open source software which:
- Provides efficient handling of large sparse tensors.
Goals and Contributions

- Invent an analysis technique which models:
  - Text Document Influencing Factors
  - Text Document Author Contributions
  - Semantics of Influences and Author Contributions

- Create open source software which:
  - Provides efficient handling of large sparse tensors.
  - Allows binding to high level languages.
Invent an analysis technique which models:
- Text Document Influencing Factors
- Text Document Author Contributions
- Semantics of Influences and Author Contributions

Create open source software which:
- Provides efficient handling of large sparse tensors.
- Allows binding to high level languages.
- Uses MPI to decompose very large sparse tensors.
  (partially completed)
Related Work I

- **Frequency Counting and Attribution**
  - *All the way through: testing for authorship in different frequency strata*. John Burrows. 2006 [2]

- **n-gram attribution**
Related Work II

- Tensors and Decompositions
  - *Tensor Decompositions and Applications*. Tamara Kolda and Brett Bader. 2009 [10]
Tensors are a generalization of matrices.
Introduction to Tensors

- Tensors are a generalization of matrices.
- The number of *modes* of a tensor is the number of indices needed to address the tensor elements.

A $4 \times 4 \times 3$ Tensor
Introduction to Tensors

- Tensors are a generalization of matrices.
- The number of *modes* of a tensor is the number of indices needed to address the tensor elements.
  - **Scalar**: 0 modes

A $4 \times 4 \times 3$ Tensor
Tensors are a generalization of matrices.

The number of *modes* of a tensor is the number of indices needed to address the tensor elements.
- **scalar** 0 modes
- **vector** 1 mode

A $4 \times 4 \times 3$ Tensor
Introduction to Tensors

- Tensors are a generalization of matrices.
- The number of *modes* of a tensor is the number of indices needed to address the tensor elements.
  - **scalar** 0 modes
  - **vector** 1 mode
  - **matrix** 2 modes

A $4 \times 4 \times 3$ Tensor
Tensors are a generalization of matrices.

The number of modes of a tensor is the number of indices needed to address the tensor elements.

- **Scalar**: 0 modes
- **Vector**: 1 mode
- **Matrix**: 2 modes
- **Tensor**: > 2 modes

A $4 \times 4 \times 3$ Tensor
Tensor Decomposition

- First studied by Frank Hitchcock in 1927 [7]
Tensor Decomposition

- First studied by Frank Hitchcock in 1927 [7]
Tensor Decomposition

- First studied by Frank Hitchcock in 1927 [7]
- The polyadic form of a tensor

\[ T \approx \sum_{i=1}^{r} a_i \otimes b_i \otimes c_i \]
Tensor Decomposition

- First studied by Frank Hitchcock in 1927 [7]
- The polyadic form of a tensor
  \[ T \approx \sum_{i=1}^{r} a_i \otimes b_i \otimes c_i \]
- Normalized polyadic form
  \[ T \approx \sum_{i=1}^{r} \lambda_i a'_i \otimes b'_i \otimes c'_i \]
Other Decomposition Techniques

- Tucker Decomposition (Kolda 2009) [10]

\[ T \approx G \times_1 A \times_2 B \times_3 C \]
Other Decomposition Techniques

- Tucker Decomposition (Kolda 2009) [10]

\[ \mathcal{T} \approx \mathbf{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \]

- Tucker Decomposition (element-wise formulation) (Kolda 2009) [10]

\[ t_{ijk} \approx \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} a_{ip} b_{jq} c_{kr} \]
Other Decomposition Techniques

- Tucker Decomposition (Kolda 2009) [10]
  \[ \mathcal{T} \approx G \times_1 A \times_2 B \times_3 C \]

- Tucker Decomposition (element-wise formulation) (Kolda 2009) [10]
  \[ t_{ijk} \approx \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} a_{ip} b_{jq} c_{kr} \]

- Non-Negative Decomposition
Properties of Tensor Decomposition

- Decompositions are hierarchical (Kiers 1991) [9].

Decompositions are hierarchical (Kiers 1991) [9].
Properties of Tensor Decomposition

- Decompositions are hierarchical (Kiers 1991) [9].
- Polyadic decomposition is unique under rotation.
Properties of Tensor Decomposition

- Decompositions are hierarchical (Kiers 1991) [9].
- Polyadic decomposition is unique under rotation.
- Tensor decompositions retain structure.
Properties of Tensor Decomposition

- Decompositions are hierarchical (Kiers 1991) [9].
- Polyadic decomposition is unique under rotation.
- Tensor decompositions retain structure.
- Normalized polyadic decompositions provide proportional profiles (Harshman 1970) [6]
Outline

1. Introduction
   - Problem Statement
   - Background

2. Approach
   - Model Overview
   - Implementation

3. Results
   - A Simple Example
   - Analysis of a Conference Paper
Let $V$ be the set of all unique words in a corpus.
Let $V$ be the set of all unique words in a corpus.

Construct an $n$ mode tensor $D \in \mathbb{R}^{|V| \times \cdots \times |V|}$
Let $V$ be the set of all unique words in a corpus.

Construct an $n$ mode tensor $\mathcal{D} \in \mathbb{R}^{|V| \times \ldots \times |V|}$

Entry $d_{ijk}$ in $\mathcal{D}$ counts the frequency of the $n$-gram word$_i$, word$_j$, word$_k$
Representing Documents as Tensors

- Let $V$ be the set of all unique words in a corpus.
- Construct an $n$ mode tensor $D \in \mathbb{R}^{|V| \times \ldots \times |V|}$
- Entry $d_{ijk}$ in $D$ counts the frequency of the $n$-gram word$_i$, word$_j$, word$_k$
- $D$ counts the frequency of every possible $n$-gram over the vocabulary $V$
Each document tensor is broken into factors using non-negative polyadic decomposition

\[ D = \sum F_i \]
Non-Negative Decomposition of Document Tensors

- Each document tensor is broken into factors using non-negative polyadic decomposition
  \[ D = \sum F_i \]
- Each factor is normalized using the $L_1$ norm.
  \[ D = \sum \lambda_i F_i' \]
Non-Negative Decomposition of Document Tensors

- Each document tensor is broken into factors using non-negative polyadic decomposition

\[ D = \sum F_i \]

- Each factor is normalized using the $L_1$ norm.

\[ D = \sum \lambda_i F'_i \]

- Each normalized factor is a proportional profile of the frequencies of $n$-grams within each document.
Non-Negative Decomposition of Document Tensors

- Each document tensor is broken into factors using non-negative polyadic decomposition

\[ D = \sum F_i \]

- Each factor is normalized using the $L_1$ norm.

\[ D = \sum \lambda_i F'_i \]

- Each normalized factor is a proportional profile of the frequencies of $n$-grams within each document.

- $\lambda_i$ expresses the importance of the factor to the document.
Matching Document Components

- Let $C$ be a corpus of document tensors.
Matching Document Components

- Let $C$ be a corpus of document tensors.
- Let $D_t \in C$ be the target document.
Matching Document Components

- Let $C$ be a corpus of document tensors.
- Let $D_t \in C$ be the target document.
- The set $C - D_t$ is the set of source documents.
Matching Document Components

- Let \( C \) be a corpus of document tensors.
- Let \( D_t \in C \) be the target document.
- The set \( C - D_t \) is the set of source documents.
- Each source document \( s \) decomposes into \( F'_s \) and \( \Lambda_s \).
Let \( C \) be a corpus of document tensors.
Let \( D_t \in C \) be the target document.
The set \( C - D_t \) is the set of source documents.
Each source document \( s \) decomposes into \( F'_s \) and \( \Lambda_s \).
The target document decomposes into \( F'_t \) and \( \Lambda_t \).
Let $C$ be a corpus of document tensors.

Let $D_t \in C$ be the target document.

The set $C - D_t$ is the set of source documents.

Each source document $s$ decomposes into $F'_s$ and $\Lambda_s$.

The target document decomposes into $F'_t$ and $\Lambda_t$.

Ascribing target document factors to source factors produces the model:

$$D_t \approx \sum_{s=1}^{\mid S \mid} \lambda^s_{t} F'_s + \lambda^n_{t} F'_n$$
Influence Model

\[ D_t \approx \sum_{s=1}^{S} \lambda_t^{s} \mathcal{F}_t^{s} + \lambda_t^{n} \mathcal{F}_t^{n} \]

- Target document weights are computed from \( \Lambda_t \)

\[ W = \frac{1}{\sum \Lambda_t} \Lambda_t \]
Influence Model

\[ D_t \approx \sum_{s=1}^{\left| S \right|} \lambda_{s}^{s} \mathcal{F}_{t}^{s} + \lambda_{t}^{n} \mathcal{F}_{t}^{n} \]

- Target document weights are computed from \( \Lambda_{t} \)

\[ W = \frac{1}{\sum \Lambda_{t}} \Lambda_{t} \]

- Weights associated with factors attributed to source factors are added to the weight of their respective documents.
Influence Model

\[ D_t \approx \sum_{s=1}^{\left| S \right|} \lambda_t^s F_t^s + \lambda_t^n F_t^n \]

- Target document weights are computed from \( \Lambda_t \)

\[ W = \frac{1}{\sum \Lambda_t} \Lambda_t \]

- Weights associated with factors attributed to source factors are added to the weight of their respective documents.
- Weights associated with factors not attributed to source factors are added to the author’s contribution weight.
Overall Algorithm

input : docs, n, nfactors, threshold
output: W, S, F

prepare(docs);
V ← build_vocabulary(docs);
C ← ∅;
foreach d in docs do
    D ← build_tensor(d, n, V);
    C ← C ∪ {D};
end
Λ,F ← extract_factors(C, nfactors);
M ← build_distance_matrix(F);
λ ← the entries in Λ corresponding to the target document.;
W, S ← extract_influence(|docs|, M,F,λ, threshold);
return W, S, F;

Algorithm 1: Influence Model Construction
Corpus Preparation

input : \textit{docs}  
output: None

\textbf{foreach} \textit{d in docs} \textbf{do}
  \begin{itemize}
    \item Remove Punctuation from \textit{d};
    \item Remove Numbers from \textit{d};
    \item Convert \textit{d} to lower case;
  \end{itemize}
\textbf{end}

\textbf{Algorithm 2:} Prepare
**Vocabulary Extraction**

**Algorithm 3: Build Vocabulary**

\[
\text{input : } \textit{docs} \\
\text{output: } V
\]

\[
V \leftarrow \emptyset; \\
\text{foreach } d \text{ in } \textit{docs} \text{ do} \\
\quad \text{foreach } \textit{word} \text{ in } d \text{ do} \\
\quad \quad V \leftarrow V \cup \{\textit{word}\}; \\
\quad \text{end} \\
\text{end} \\
\text{return } V;
\]
Build Document Tensor

```
the  cat  sat  on  the  mat
the  cat  sat  on  the  mat
the  cat  sat  on  the  mat
the  cat  sat  on  the  mat
the  cat  sat  on  the  mat
```
Building Document Tensors

**input**: \(d, n, V, n\)

**output**: \(\mathcal{D}\)

\(\mathcal{D} \leftarrow \text{Tensor with dimension } |V| \times |V| \times \ldots \times n |V|;\)

Fill \(\mathcal{D}\) with 0;

\(\text{len} \leftarrow \text{number of words in } d;\)

for \(i \leftarrow 1 \text{ to } \text{len} - n\) do

/* Compute Tensor Element Index */

\(\text{index} \leftarrow \text{list of } n \text{ integers};\)

for \(j \leftarrow 1 \text{ to } n\) do

| \(\text{index}[j] \leftarrow \text{index of word } d[i] \text{ in } V;\)

end

/* Update Frequency of This \(n\)-gram */

\(\mathcal{D}[\text{index}] \leftarrow \mathcal{D}[\text{index}] + 1;\)

end

return \(\mathcal{D}\)

**Algorithm 4: Build Tensor**
Tensor Decomposition

Algorithm 5: Extract Factors

```plaintext
input : C, nfactors
output: Λ, F

F ← ∅;
Λ ← ∅;
nmodes ← number of modes in C[1];
foreach D in C do
    U ← ccd_ntfd(D, nfactors);
    foreach i = 1 to nfactors do
        T ← U[1][:, i];
        for m = 2 to nmodes do
            T ← T ⊗ U[m][:, i];
        end
        λ ← L_1_norm(T);
        T ← T / λ;
    end
    F ← F ∪ {T};
    Λ ← Λ ∪ {λ};
end
return Λ, F
```

Robert Earl Lowe  Textual Influence Modeling
Distance Computation

\[
\text{input} : F \\
\text{output: } M
\]

\[
M \leftarrow \text{Matrix with dimension } |F| \times |F|; \\
\text{for } i = 1 \text{ to } |F| \text{ do} \\
\hspace{1em} \text{for } j = 1 \text{ to } |F| \text{ do} \\
\hspace{2em} M[i, j] \leftarrow \text{L}_1\text{norm}(F[i] - F[j]); \\
\hspace{1em} \text{end} \\
\text{end} \\
\text{return } M
\]

Algorithm 6: Build Distance Matrix
Factor Matching

input : \textit{ndocs}, \textit{M}, \textit{F}, \lambda, \textit{threshold}

output: \textit{W}, \textit{S}

/* Compute Weights */
sum \leftarrow \sum \lambda;
W \leftarrow \lambda/\text{sum};
\text{S} \leftarrow \text{list of integers of size } |\lambda|;

/* Classify Factors */
nfactors \leftarrow |\lambda|;
for \textit{i} = 1 \text{ to } nfactors \text{ do}
    \text{min} \leftarrow \text{M}[\text{row}, 1];
    \text{minIndex} \leftarrow 1;
    \text{row} \leftarrow \text{i} + nfactors \times (\text{ndocs} - 1);
    for \textit{j} = 1 \text{ to } nfactors \times \text{ndocs} \text{ do}
        \text{if } \text{M}[\text{row}, \text{j}] < \text{min} \text{ then}
            \text{min} \leftarrow \text{M}[\text{row}, \text{j}];
            \text{minIndex} \leftarrow \text{j};
        \text{end}
    \text{if } \text{min} \leq \text{threshold} \text{ then}
        \text{S}[\text{i}] \leftarrow \text{minIndex};
    \text{else}
        \text{S}[\text{i}] \leftarrow 0;
    \text{end}
\text{return } \text{W}, \text{S};

\textbf{Algorithm 7: Extract Influence}
Final Summation

**input**: $ndocs$, $S$, $W$

**output**: $I$, $author$

$I \leftarrow$ List of 0 repeated $ndocs - 1$ times;

for $i = 1$ to $ndocs$ do

if $S[i] = 0$ then

$author = author + W[i]$;

else

$j \leftarrow$ Document number corresponding with $S[i]$;

$I[j] \leftarrow I[j] + W[i]$;

end

end

**Algorithm 8: Final Summation**
Tensor functions are implemented as an ANSI C library called sptensor.
Implementation Details

- Tensor functions are implemented as an ANSI C library called sptensor.
- The document influence model is implemented as a series of C programs and shell scripts. Each algorithm is a standalone program.
Tensor functions are implemented as an ANSI C library called sptensor.

The document influence model is implemented as a series of C programs and shell scripts. Each algorithm is a standalone program.

Because the MPI version of sptensor is not yet complete, vocabularies are constrained to a maximum of 600 words.
Tensor functions are implemented as an ANSI C library called sptensor.

The document influence model is implemented as a series of C programs and shell scripts. Each algorithm is a standalone program.

Because the MPI version of sptensor is not yet complete, vocabularies are constrained to a maximum of 600 words.

- Sort the vocabulary by frequency.
Implementation Details

- Tensor functions are implemented as an ANSI C library called sptensor.
- The document influence model is implemented as a series of C programs and shell scripts. Each algorithm is a standalone program.
- Because the MPI version of sptensor is not yet complete, vocabularies are constrained to a maximum of 600 words.
  - Sort the vocabulary by frequency.
  - Keep the 599 most frequent words.
Implementation Details

- Tensor functions are implemented as an ANSI C library called sptensor.
- The document influence model is implemented as a series of C programs and shell scripts. Each algorithm is a standalone program.
- Because the MPI version of sptensor is not yet complete, vocabularies are constrained to a maximum of 600 words.
  - Sort the vocabulary by frequency.
  - Keep the 599 most frequent words.
  - Insert a new symbol, @, to act as a wildcard.
Tensor functions are implemented as an ANSI C library called sptensor.

The document influence model is implemented as a series of C programs and shell scripts. Each algorithm is a standalone program.

Because the MPI version of sptensor is not yet complete, vocabularies are constrained to a maximum of 600 words.

- Sort the vocabulary by frequency.
- Keep the 599 most frequent words.
- Insert a new symbol, @, to act as a wildcard.
- When building document tensors, all words not in the vocabulary are replaced with the wildcard.
Outline

1. Introduction
   - Problem Statement
   - Background

2. Approach
   - Model Overview
   - Implementation

3. Results
   - A Simple Example
   - Analysis of a Conference Paper
A Simple Example: Cat and Dog

The Cat’s Tale
The cat sat on the mat. The cat was happy to be on the mat. The cat saw the mouse running but was too lazy to chase it.
A Simple Example: Cat and Dog

The Cat’s Tale
The cat sat on the mat. The cat was happy to be on the mat. The cat saw the mouse running but was too lazy to chase it.

The Dog’s Tale
The dog walked to the house. The dog saw the food bowl, and the dog saw a squirrel. The dog chased the squirrel from the food bowl.
A Simple Example: Cat and Dog

The Cat’s Tale
The cat sat on the mat. The cat was happy to be on the mat. The cat saw the mouse running but was too lazy to chase it.

The Dog’s Tale
The dog walked to the house. The dog saw the food bowl, and the dog saw a squirrel. The dog chased the squirrel from the food bowl.

The Saga Continues
The dog saw the cat on the mat. The dog walked to the house, and the dog chased the cat. The squirrel was happy to see the dog chase the cat on the mat. The dog saw the squirrel, and decided to chase the squirrel instead. The cat sat on the mat.
Cat and Dog Vocabulary and Tensors

<table>
<thead>
<tr>
<th>Vocabulary</th>
<th>Non-Zero Entries of Cat Tensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Word</td>
</tr>
<tr>
<td>1</td>
<td>the</td>
</tr>
<tr>
<td>2</td>
<td>house</td>
</tr>
<tr>
<td>3</td>
<td>mouse</td>
</tr>
<tr>
<td>4</td>
<td>squirrel</td>
</tr>
<tr>
<td>5</td>
<td>it</td>
</tr>
<tr>
<td>6</td>
<td>saw</td>
</tr>
<tr>
<td>7</td>
<td>lazy</td>
</tr>
<tr>
<td>8</td>
<td>cat</td>
</tr>
<tr>
<td>9</td>
<td>mat</td>
</tr>
<tr>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>11</td>
<td>bowl</td>
</tr>
<tr>
<td>12</td>
<td>walked</td>
</tr>
<tr>
<td>13</td>
<td>too</td>
</tr>
<tr>
<td>14</td>
<td>and</td>
</tr>
<tr>
<td>15</td>
<td>see</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cat and Dog Model Parameters and Output

<table>
<thead>
<tr>
<th>Model Parameters</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>n-gram size</td>
<td>3</td>
</tr>
<tr>
<td>nfactors</td>
<td>7</td>
</tr>
<tr>
<td>threshold</td>
<td>0.2</td>
</tr>
<tr>
<td>Corpus Size</td>
<td>3</td>
</tr>
<tr>
<td>Total Word Count</td>
<td>107</td>
</tr>
<tr>
<td>Corpus Sparsity</td>
<td>99.7%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model Output</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor</td>
<td>Factor Weight</td>
</tr>
<tr>
<td>1</td>
<td>0.28</td>
</tr>
<tr>
<td>2</td>
<td>0.15</td>
</tr>
<tr>
<td>3</td>
<td>0.14</td>
</tr>
<tr>
<td>4</td>
<td>0.14</td>
</tr>
<tr>
<td>5</td>
<td>0.11</td>
</tr>
<tr>
<td>6</td>
<td>0.11</td>
</tr>
<tr>
<td>7</td>
<td>0.06</td>
</tr>
</tbody>
</table>

Author Contribution 0.79
Cat Contribution 0.15
Dog Contribution 0.06
# A Simple Example

## Analysis of a Conference Paper

### Cat and Dog Influencing Factors

<table>
<thead>
<tr>
<th>Matched to Cat Factor 1</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 1</td>
<td>Word 2</td>
<td>Word 3</td>
<td>Proportion</td>
</tr>
<tr>
<td>on</td>
<td>the</td>
<td>mat</td>
<td>1.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Matched to Dog Factor 1</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 1</td>
<td>Word 2</td>
<td>Word 3</td>
<td>Proportion</td>
</tr>
<tr>
<td>the</td>
<td>dog</td>
<td>saw</td>
<td>0.40</td>
</tr>
<tr>
<td>the</td>
<td>dog</td>
<td>walked</td>
<td>0.20</td>
</tr>
<tr>
<td>the</td>
<td>dog</td>
<td>chased</td>
<td>0.20</td>
</tr>
<tr>
<td>the</td>
<td>dog</td>
<td>chase</td>
<td>0.20</td>
</tr>
</tbody>
</table>
## A Simple Example

### Analysis of a Conference Paper

**Cat and Dog Original Factors**

<table>
<thead>
<tr>
<th>Word 1</th>
<th>Word 2</th>
<th>Word 3</th>
<th>Proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>saw</td>
<td>the</td>
<td>squirrel</td>
<td>0.267417</td>
</tr>
<tr>
<td>saw</td>
<td>the</td>
<td>cat</td>
<td>0.223651</td>
</tr>
<tr>
<td>saw</td>
<td>the</td>
<td>dog</td>
<td>0.192194</td>
</tr>
<tr>
<td>cat</td>
<td>the</td>
<td>squirrel</td>
<td>0.044066</td>
</tr>
<tr>
<td>cat</td>
<td>the</td>
<td>cat</td>
<td>0.036854</td>
</tr>
<tr>
<td>cat</td>
<td>the</td>
<td>dog</td>
<td>0.031670</td>
</tr>
<tr>
<td>mat</td>
<td>the</td>
<td>squirrel</td>
<td>0.034331</td>
</tr>
<tr>
<td>mat</td>
<td>the</td>
<td>cat</td>
<td>0.028712</td>
</tr>
<tr>
<td>mat</td>
<td>the</td>
<td>dog</td>
<td>0.024674</td>
</tr>
<tr>
<td>see</td>
<td>the</td>
<td>squirrel</td>
<td>0.032132</td>
</tr>
<tr>
<td>see</td>
<td>the</td>
<td>cat</td>
<td>0.026873</td>
</tr>
<tr>
<td>see</td>
<td>the</td>
<td>dog</td>
<td>0.023094</td>
</tr>
<tr>
<td>chased</td>
<td>the</td>
<td>squirrel</td>
<td>0.013437</td>
</tr>
<tr>
<td>chased</td>
<td>the</td>
<td>cat</td>
<td>0.011238</td>
</tr>
<tr>
<td>chased</td>
<td>the</td>
<td>dog</td>
<td>0.009657</td>
</tr>
<tr>
<td>squirrel and happy</td>
<td></td>
<td>0.249836</td>
<td></td>
</tr>
<tr>
<td>squirrel and decided</td>
<td></td>
<td>0.262960</td>
<td></td>
</tr>
<tr>
<td>squirrel was happy</td>
<td></td>
<td>0.237368</td>
<td></td>
</tr>
<tr>
<td>squirrel was decided</td>
<td></td>
<td>0.249836</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word 1</th>
<th>Word 2</th>
<th>Word 3</th>
<th>Proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>decided</td>
<td>to</td>
<td>chase</td>
<td>1.000000</td>
</tr>
<tr>
<td>happy</td>
<td>to</td>
<td>see</td>
<td>1.000000</td>
</tr>
<tr>
<td>cat</td>
<td>saw</td>
<td>the</td>
<td>0.345830</td>
</tr>
<tr>
<td>cat</td>
<td>see</td>
<td>the</td>
<td>0.040819</td>
</tr>
<tr>
<td>cat</td>
<td>chased</td>
<td>the</td>
<td>0.172914</td>
</tr>
<tr>
<td>cat</td>
<td>chase</td>
<td>the</td>
<td>0.213734</td>
</tr>
<tr>
<td>walked</td>
<td>saw</td>
<td>the</td>
<td>0.056987</td>
</tr>
<tr>
<td>walked</td>
<td>see</td>
<td>the</td>
<td>0.006726</td>
</tr>
<tr>
<td>walked</td>
<td>chased</td>
<td>the</td>
<td>0.028493</td>
</tr>
<tr>
<td>walked</td>
<td>chase</td>
<td>the</td>
<td>0.035220</td>
</tr>
<tr>
<td>to</td>
<td>saw</td>
<td>the</td>
<td>0.044398</td>
</tr>
<tr>
<td>to</td>
<td>see</td>
<td>the</td>
<td>0.005240</td>
</tr>
<tr>
<td>to</td>
<td>chased</td>
<td>the</td>
<td>0.022199</td>
</tr>
<tr>
<td>to</td>
<td>chase</td>
<td>the</td>
<td>0.027439</td>
</tr>
</tbody>
</table>
Corpus of Scientific Papers

<table>
<thead>
<tr>
<th></th>
<th>Title</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A symbolic representation of time series, with implications for streaming algorithms.</td>
<td>Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. ACM Press, 2003</td>
</tr>
<tr>
<td>2</td>
<td>Using hmm based recognizers for writer identification and verification.</td>
<td>Andreas Schlapbach and Horst Bunke. IEEE, 2004</td>
</tr>
<tr>
<td>5</td>
<td>Multilinear operators for higher-order decompositions.</td>
<td>Kolda, Tamara Gibson. 2006</td>
</tr>
<tr>
<td>6</td>
<td>Latent dirichlet allocation.</td>
<td>Blei, David M and Ng, Andrew Y and Jordan, Michael I. 2007</td>
</tr>
</tbody>
</table>
## Model Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n)-gram size</td>
<td>3</td>
</tr>
<tr>
<td>( nfactors )</td>
<td>150</td>
</tr>
<tr>
<td>( threshold )</td>
<td>0.2</td>
</tr>
<tr>
<td>Corpus Size</td>
<td>7</td>
</tr>
<tr>
<td>Total Word Count</td>
<td>45,152</td>
</tr>
<tr>
<td>Corpus Sparsity</td>
<td>99.993%</td>
</tr>
</tbody>
</table>
Information From Reading the Target Paper

- The first cited source details the algorithm which the author extends. The factors pulled from this source all discuss the properties of the original algorithm.

<table>
<thead>
<tr>
<th>Document</th>
<th>Influence</th>
<th>Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.21</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>0.09</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>0.06</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>0.06</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>Author</td>
<td>0.57</td>
<td>127</td>
</tr>
</tbody>
</table>
Information From Reading the Target Paper

- The first cited source details the algorithm which the author extends. The factors pulled from this source all discuss the properties of the original algorithm.
- The second, third, and fourth cited sources are previous algorithms, to which the new one is compared.
Information From Reading the Target Paper

- The first cited source details the algorithm which the author extends. The factors pulled from this source all discuss the properties of the original algorithm.
- The second, third, and fourth cited sources are previous algorithms, to which the new one is compared.
- Papers five and six are from a completely unrelated field.
Introduction

Approach

Results

A Simple Example

Analysis of a Conference Paper

Distribution of All Factor Distances
Distribution of Target Factor Distances
Conclusion and Future Work

- Non-Negative Tensor Factorization can be used to build an influence model of text documents.
Non-Negative Tensor Factorization can be used to build an influence model of text documents.
Semantic information extracted from the model matches expectations.
Conclusion and Future Work

- Non-Negative Tensor Factorization can be used to build an influence model of text documents.
- Semantic information extracted from the model matches expectations.
- Future Research Directions
Conclusion and Future Work

- Non-Negative Tensor Factorization can be used to build an influence model of text documents.
- Semantic information extracted from the model matches expectations.
- Future Research Directions
  - Complete the MPI implementation of sptensor
Conclusion and Future Work

- Non-Negative Tensor Factorization can be used to build an influence model of text documents.
- Semantic information extracted from the model matches expectations.
- Future Research Directions
  - Complete the MPI implementation of sptensor
  - Replicate the Burrows and Craig 2017 study of Henry VI, part 3
Conclusion and Future Work

- Non-Negative Tensor Factorization can be used to build an influence model of text documents.
- Semantic information extracted from the model matches expectations.
- Future Research Directions
  - Complete the MPI implementation of sptensor
  - Replicate the Burrows and Craig 2017 study of Henry VI, part 3
  - Study the effects of constraining the vocabulary
Conclusion and Future Work

- Non-Negative Tensor Factorization can be used to build an influence model of text documents.
- Semantic information extracted from the model matches expectations.
- Future Research Directions
  - Complete the MPI implementation of sptensor
  - Replicate the Burrows and Craig 2017 study of Henry VI, part 3
  - Study the effects of constraining the vocabulary
  - Apply the model to identify possible chronologies in documents where chronology and provenance are questioned.
Conclusion and Future Work

- Non-Negative Tensor Factorization can be used to build an influence model of text documents.
- Semantic information extracted from the model matches expectations.
- Future Research Directions
  - Complete the MPI implementation of sptensor
  - Replicate the Burrows and Craig 2017 study of Henry VI, part 3
  - Study the effects of constraining the vocabulary
  - Apply the model to identify possible chronologies in documents where chronology and provenance are questioned.
  - Use the model to build a network of influence flow in a hierarchical corpus.
Acknowledgments

I would like to thank

- My advisor Dr. Mike Berry
- My committee Dr. Audris Mockus, Dr. Brad Vander Zanden, Dr. Judy Day
- Graduate Student Administrator Ms. Dana Bryson
- All of my colleagues at Maryville College
- My Wife Erin Lowe


*Sheakespeare, Computers, and the Mystery of Authorship.*  

Foundations of the parafac procedure: Models and conditions for an" explanatory" multi-modal factor analysis.  
1970.

