Textual Influence Modeling Through Non-Negative Tensor Decomposition

Robert Earl Lowe

July 12, 2018

THE UNIVERSITY OF TENNESSEE KNOXVILLE

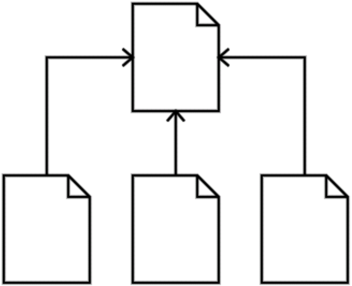
Outline

1. Introduction
   - Problem Statement
   - Background
2. Approach
   - Model Overview
   - Implementation
3. Results
   - A Simple Example
   - Analysis of a Conference Paper

Outline

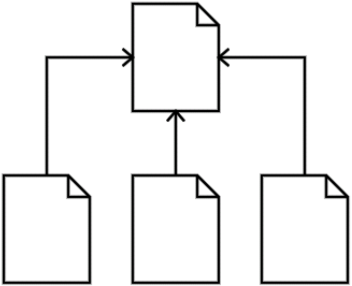
Text Documents and Influences

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



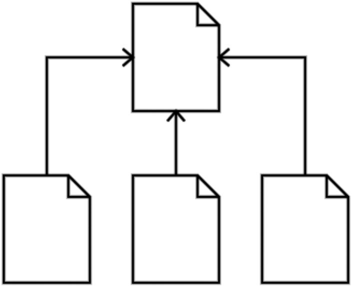
Text Documents and Influences

- Every text document is a combination of an author's contributions and contributing factors.
- Contributing Factors



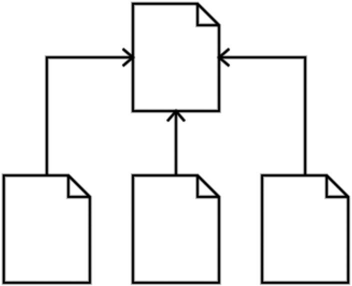
Text Documents and Influences

- Every text document is a combination of an author's contributions and contributing factors.
- Contributing Factors
  - Cited Sources



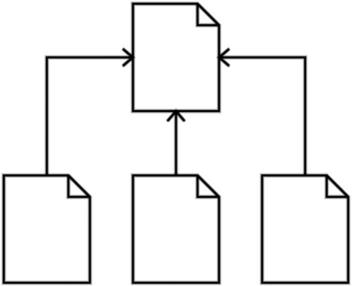
Text Documents and Influences

- 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:

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:

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.

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.

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. (partially completed)

Related Work I

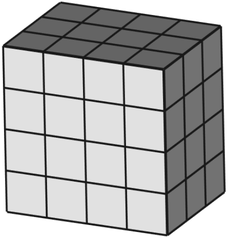
- Frequency Counting and Attribution
  - *All the way through: testing for authorship in different frequency strata*. John Burrows. 2006 [2]
  - *The Joker in the Pack?: Marlowe, Kyd, and the Co-authorship of Henry VI, Part 3*. John Burrows and Hugh Craig. 2017 [3]
  - *Sheakespeare, Computers, and the Mystery of Authorship*. Hugh Craig and Arthur Kinney. 2009 [5]
- *n*-gram attribution
  - *N-gram over Context*. Noriaki Kawamae. 2016 [8]
  - *Language chunking, data sparseness, and the value of a long marker list: explorations with word n-grams and authorial attribution*. Alexis Antonia, Hugh Craig, and Jack Elliott. 2014 [1]

Related Work II

- Tensors and Decompositions
  - *Tensor Decompositions and Applications*. Tamara Kolda and Brett Bader. 2009 [10]
  - *Foundations of the PARAFAC procedure: Models and conditions for ani "explanatory" multi-modal factor analysis*. Richard Harshman. 1970 [6]
  - *Sparse non-negative tensor factorization using columnwise coordinate descent*. Ji Liu, Jun Liu, Peter Wonka, and Jieping Yi. 2012[11]

Introduction to Tensors

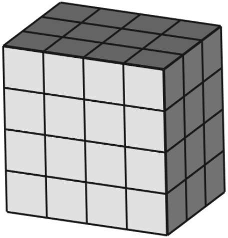
- Tensors are a generalization of matrices.



A $4 \times 4 \times 3$ Tensor

Introduction to Tensors

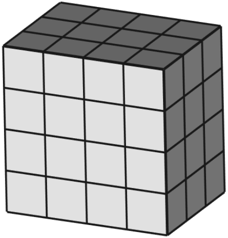
- Tensors are a generalization of matrices.
- The number of *modes* of a tensor is the number of indices needed to address the tensor elements.



A $4 \times 4 \times 3$ Tensor

Introduction to Tensors

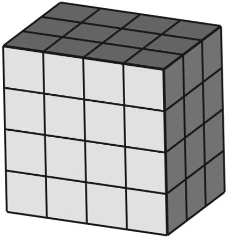
- Tensors are a generalization of matrices.
- The number of *modes* of a tensor is the number of indices needed to address the tensor elements.
    - **scalar** 0 modes



A $4 \times 4 \times 3$ Tensor

Introduction to Tensors

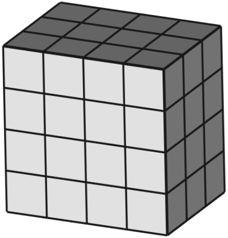
- Tensors are a generalization of matrices.
- The number of *modes* of a tensor is the number of indices needed to address the tensor elements.
  - **scalar** 0 modes
  - **vector** 1 mode



A $4 \times 4 \times 3$ Tensor

Introduction to Tensors

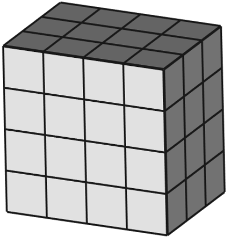
- Tensors are a generalization of matrices.
- The number of *modes* of a tensor is the number of indices needed to address the tensor elements.
  - **scalar** 0 modes
  - **vector** 1 mode
  - **matrix** 2 modes



A $4 \times 4 \times 3$ Tensor

Introduction to Tensors

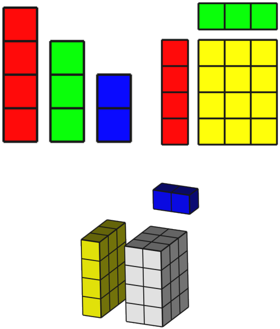
- Tensors are a generalization of matrices.
- The number of *modes* of a tensor is the number of indices needed to address the tensor elements.
  - **scalar** 0 modes
  - **vector** 1 mode
  - **matrix** 2 modes
  - **tensor** $>$ 2 modes



A $4 \times 4 \times 3$ Tensor

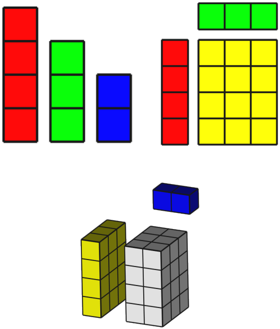
Tensor Decomposition

- First studied by Frank Hitchcock in 1927 [7]



Tensor Decomposition

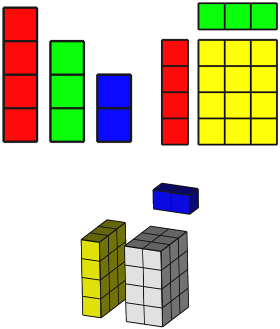
- First studied by Frank Hitchcock in 1927 [7]
- Popularized by Richard Harshman [6] and Carroll and Chang [4] in the 1970's.



Tensor Decomposition

- First studied by Frank Hitchcock in 1927 [7]
- Popularized by Richard Harshman [6] and Carroll and Chang [4] in the 1970's.
- The polyadic form of a tensor

$$\mathcal{T} \approx \sum_{i=1}^{r} a_i \otimes b_i \otimes c_i$$



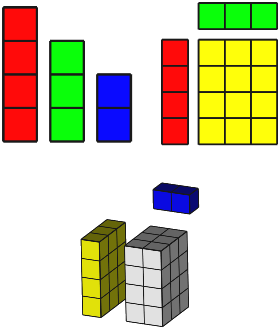
Tensor Decomposition

- First studied by Frank Hitchcock in 1927 [7]
- Popularized by Richard Harshman [6] and Carroll and Chang [4] in the 1970's.
- The polyadic form of a tensor

$$\mathcal{T} \approx \sum_{i=1}^{r} a_i \otimes b_i \otimes c_i$$

- Normalized polyadic form

$$\mathcal{T} \approx \sum_{i=1}^{r} \lambda_i a_i' \otimes b_i' \otimes c_i'$$



Other Decomposition Techniques

- Tucker Decomposition (Kolda 2009) [10]

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

Other Decomposition Techniques

- Tucker Decomposition (Kolda 2009) [10]

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

- 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]

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

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

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

- Non-Negative Decomposition

Properties of Tensor Decomposition

- 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

Representing Documents as Tensors

- Let V be the set of all unique words in a corpus.

Representing Documents as Tensors

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

Representing Documents as Tensors

- Let V be the set of all unique words in a corpus.
- Construct an *n* mode tensor $\mathcal{D} \in \mathbb{R}^{|\mathrm{V}| \times \ldots \times |\mathrm{V}|}$
- Entry d_{ijk} in $\mathcal{D}$ counts the frequency of the *n*-gram $\mathrm{word}_i, \mathrm{word}_j, \mathrm{word}_k$

Representing Documents as Tensors

- Let V be the set of all unique words in a corpus.
- Construct an n mode tensor $\mathcal{D} \in \mathbb{R}^{|\mathrm{V}| \times \ldots \times |\mathrm{V}|}$
- Entry d_{ijk} in $\mathcal{D}$ counts the frequency of the n-gram $\mathrm{word}_i, \mathrm{word}_j, \mathrm{word}_k$
- $\mathcal{D}$ counts the frequency of every possible n-gram over the vocabulary V

Non-Negative Decomposition of Document Tensors

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

$$\mathcal{D} = \sum \mathcal{F}_i$$

Non-Negative Decomposition of Document Tensors

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

$$\mathcal{D} = \sum \mathcal{F}_i$$

- Each factor is normalized using the L_1 norm.

$$\mathcal{D} = \sum \lambda_i \mathcal{F}'_i$$

Non-Negative Decomposition of Document Tensors

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

$$\mathcal{D} = \sum \mathcal{F}_i$$

- Each factor is normalized using the L_1 norm.

$$\mathcal{D} = \sum \lambda_i \mathcal{F}'_i$$

- 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

$$\mathcal{D} = \sum \mathcal{F}_i$$

- Each factor is normalized using the L_1 norm.

$$\mathcal{D} = \sum \lambda_i \mathcal{F}'_i$$

- Each normalized factor is a proportional profile of the frequencies of n-grams within each document.
- λ_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 $\mathcal{D}_t \in \mathrm{C}$ be the target document.

Matching Document Components

- Let C be a corpus of document tensors.
- Let $\mathcal{D}_t \in \mathrm{C}$ be the target document.
- The set $\mathrm{C} - \mathcal{D}_t$ is the set of source documents.

Matching Document Components

- Let C be a corpus of document tensors.
- Let $\mathcal{D}_t \in \mathrm{C}$ be the target document.
- The set $\mathrm{C} - \mathcal{D}_t$ is the set of source documents.
- Each source document s decomposes into $\mathrm{F'}_s$ and Λ_s.

Matching Document Components

- Let C be a corpus of document tensors.
- Let $\mathcal{D}_t \in \mathrm{C}$ be the target document.
- The set $\mathrm{C} - \mathcal{D}_t$ is the set of source documents.
- Each source document s decomposes into $\mathrm{F'}_s$ and Λ_s.
- The target document decomposes into $\mathrm{F'}_t$ and Λ_t

Matching Document Components

- Let C be a corpus of document tensors.
- Let $\mathcal{D}_t \in \mathrm{C}$ be the target document.
- The set $\mathrm{C} - \mathcal{D}_t$ is the set of source documents.
- Each source document s decomposes into F'_s and Λ_s.
- The target document decomposes into F'_t and Λ_t
- Ascribing target document factors to source factors produces the model:

$$\mathcal{D}_t \approx \sum_{s=1}^{|\mathrm{S}|} \lambda_t^s \mathcal{F}_t'^s + \lambda_t^n \mathcal{F}_t'^n$$

Influence Model

$$\mathcal{D}_t \approx \sum_{s=1}^{|\mathrm{S}|} \lambda_t^s \mathcal{F}_t'^s + \lambda_t^n \mathcal{F}_t'^n$$

- Target document weights are computed from Λ_t

$$\mathrm{W} = \frac{1}{\sum \Lambda_t} \Lambda_t$$

Influence Model

$$\mathcal{D}_t \approx \sum_{s=1}^{|S|} \lambda_t^s \mathcal{F}_t^{\prime s} + \lambda_t^n \mathcal{F}_t^{\prime n}$$

- Target document weights are computed from Λ_t

$$\mathrm{W} = \frac{1}{\sum \Lambda_t} \Lambda_t$$

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

Influence Model

$$\mathcal{D}_t \approx \sum_{s=1}^{|\mathrm{S}|} \lambda_t^s \mathcal{F}_t'^s + \lambda_t^n \mathcal{F}_t'^n$$

- Target document weights are computed from Λ_t

$$\mathrm{W} = \frac{1}{\sum \Lambda_t} \Lambda_t$$

- 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 $\leftarrow$ `build_vocabulary`(*docs*);
C $\leftarrow \emptyset$;
foreach *d in docs* **do**
  $\mathcal{D} \leftarrow$ `build_tensor` (*d*, *n*, V);
  C $\leftarrow$ C $\cup \{\mathcal{D}\}$;
end
Λ,F $\leftarrow$ `extract_factors`(C, *nfactors*);
M $\leftarrow$ `build_distance_matrix`(F);
$\lambda \leftarrow$ the entries in Λ corresponding to the target document.;
W, S $\leftarrow$ `extract_influence`($|$*docs*$|$, *M*,F,λ, *threshold*);
return W, S, F;

Algorithm 1: Influence Model Construction

Corpus Preparation

input : *docs*
output: None

foreach *d in docs* **do**
    Remove Punctuation from *d*;
    Remove Numbers from *d*;
    Convert *d* to lower case;
end

Algorithm 2: Prepare

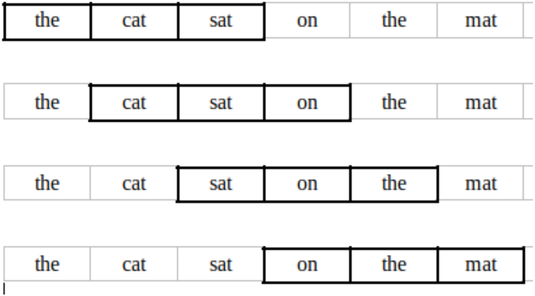
Vocabulary Extraction

input : *docs*
output: V

$\mathrm{V} \leftarrow \emptyset$;
foreach *d in docs* **do**
    foreach *word in d* **do**
        $\mathrm{V} \leftarrow \mathrm{V} \cup \{word\}$;
    end
end
return V;

Algorithm 3: Build Vocabulary

Build Document Tensor



Building Document Tensors

```
input  : d, n, V, n
output: 𝒟

𝒟 ← Tensor with dimension |V| × |V| ... ×_n |V|;
Fill 𝒟 with 0;
len ← number of words in d;
for i ← 1 to len − n do
    /* Compute Tensor Element Index                */
    index ← list of n integers;
    for j ← 1 to n do
        index[j] ← index of word d[i] in V;
    end
    /* Update Frequency of This n-gram             */
    𝒟[index] ← 𝒟[index] + 1;
end
return 𝒟
```

Algorithm 4: Build Tensor

Tensor Decomposition

```
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);
    for i = 1 to nfactors do
        /* Build the Factor                                        */
        𝒯 ← U[1][:, i];
        for m = 2 to nmodes do
            𝒯 ← 𝒯 ⊗ U[m][:, i];
        end
        /* Compute the norm and normalize the factor               */
        λ ← L_1_norm(𝒯);
        𝒯 ← 𝒯/λ;
        /* Insert the factor and norm into the list                */
        F ← F ∪ {𝒯};
        Λ ← Λ ∪ {λ};
    end
end
return Λ, F
```

Algorithm 5: Extract Factors

Distance Computation

input : F
output: M

$M \leftarrow$ Matrix with dimension $|\mathrm{F}| \times |\mathrm{F}|$;
for $i = 1$ *to* $|\mathrm{F}|$ **do**
  for $j = 1$ *to* $|\mathrm{F}|$ **do**
    $M[i,j] \leftarrow \mathrm{L}_1\_\mathtt{norm}\,(\mathrm{F}[i] - \mathrm{F}[j])$;
  end
end
return M

Algorithm 6: Build Distance Matrix

Factor Matching

```
input  : ndocs, M, F, λ, threshold
output: W, S

/* Compute Weights                                   */
sum ← ∑ λ;
W ← λ/sum;
S ← list of integers of size |λ|;
/* Classify Factors                                  */
nfactors ← |λ|;
for i = 1 to nfactors do
    min ← M[row, 1];
    minIndex ← 1;
    row ← i + nfactors ∗ (ndocs − 1);
    for j = 1 to nfactors ∗ ndocs do
        if M [row,j]< min then
            min ← M[row, j];
            minIndex ← j;
        end
    end
    if min ≤ threshold then
        S[i] ← minIndex;
    else
        S[i] ← 0;
    end
end
return W, S;
```

Algorithm 7: Extract Influence

Final Summation

input : *ndocs*, S, W
output: I, *author*

$\mathrm{I} \leftarrow$ List of 0 repeated *ndocs* $-$ 1 times;
for *i = 1 to ndocs* **do**
  if $\mathrm{S}[i] = 0$ **then**
    author $=$ *author* $+ \mathrm{W}[i]$;
  else
    $j \leftarrow$ Document number corresponding with $\mathrm{S}[i]$;
    $\mathrm{I}[j] \leftarrow \mathrm{I}[j] + \mathrm{W}[i]$;
  end
end

Algorithm 8: Final Summation

Implementation Details

- 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.

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.

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.

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.

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.
  - When building document tensors, all words not in the vocabulary are replaced with the wildcard.

Outline

A Simple Example: Cat and Dog

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

Vocabulary

I	Word	*I*	Word
1	the	16	chased
2	house	17	sat
3	mouse	18	be
4	squirrel	19	happy
5	it	20	on
6	saw	21	from
7	lazy	22	food
8	cat	23	decided
9	mat	24	to
10	a	25	was
11	bowl	26	dog
12	walked	27	running
13	too	28	instead
14	and	29	but
15	see	30	chase

Non-Zero Entries of Cat Tensor

i	*j*	*k*	*freq*
1	8	17	1
8	17	20	1
17	20	1	1
20	1	9	2
1	9	1	2
9	1	8	2
1	8	25	1
8	25	19	1
25	19	24	1
19	24	18	1
24	18	20	1
18	20	1	1
1	8	6	1
8	6	1	1
6	1	3	1
1	3	27	1
3	27	29	1
27	29	25	1
29	25	13	1
25	13	7	1
13	7	24	1
7	24	30	1

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Cat and Dog Model Parameters and Output

Model Parameters

n-gram size	3
nfactors	7
threshold	0.2
Corpus Size	3
Total Word Count	107
Corpus Sparsity	99.7%

Model Output

Factor	Factor Weight	Classification
1	0.28	Author Contribution
2	0.15	Cat Factor 1
3	0.14	Author Contribution
4	0.14	Author Contribution
5	0.11	Author Contribution
6	0.11	Author Contribution
7	0.06	Dog Factor 1

Author Contribution	0.79
Cat Contribution	0.15
Dog Contribution	0.06

Cat and Dog Influencing Factors

Matched to Cat Factor 1

Word 1	Word 2	Word 3	Proportion
on	the	mat	1.00

Matched to Dog Factor 1

Word 1	Word 2	Word 3	Proportion
the	dog	saw	0.40
the	dog	walked	0.20
the	dog	chased	0.20
the	dog	chase	0.20

Cat and Dog Original Factors

Word 1	Word 2	Word 3	Proportion
saw	the	squirrel	0.267417
saw	the	cat	0.223651
saw	the	dog	0.192194
cat	the	squirrel	0.044066
cat	the	cat	0.036854
cat	the	dog	0.031670
mat	the	squirrel	0.034331
mat	the	cat	0.028712
mat	the	dog	0.024674
see	the	squirrel	0.032132
see	the	cat	0.026873
see	the	dog	0.023094
chased	the	squirrel	0.013437
chased	the	cat	0.011238
chased	the	dog	0.009657
squirrel	and	happy	0.249836
squirrel	and	decided	0.262960
squirrel	was	happy	0.237368
squirrel	was	decided	0.249836

Word 1	Word 2	Word 3	Proportion
decided	to	chase	1.000000
happy	to	see	1.000000
cat	saw	the	0.345830
cat	see	the	0.040819
cat	chased	the	0.172914
cat	chase	the	0.213734
walked	saw	the	0.056987
walked	see	the	0.006726
walked	chased	the	0.028493
walked	chase	the	0.035220
to	saw	the	0.044398
to	see	the	0.005240
to	chased	the	0.022199
to	chase	the	0.027439

Case Study: Regional Conference Paper

Corpus of Scientific Papers

1	Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. ACM Press, 2003
2	Andreas Schlapbach and Horst Bunke. Usinghmm based recognizers for writer identification and verficiation. IEEE, 2004
3	Yusuke Manabe and Basabi Chakraborty. Identy detection from on-line handwriting time series. IEEE, 2008
4	Sami Gazzah and Najoua Ben Amara. Arabic handwriting texture analysis for writer identification using the dwt-lifting scheme. IEEE, 2007.
5	Kolda, Tamara Gibson. Multilinear operators for higher-order decompositions. 2006
6	Blei, David M and Ng, Andrew Y and Jordan, Michael I. Latent dirichlet allocation. 2007
7	Serfas, Doug. Dynamic Biometric Recognition of Handwritten Digits Using Symbolic Aggregate Approximation. Proceedings of the ACM Southeast Conference 2017

Model Parameters

Model Parameters	
n-gram size	3
nfactors	150
threshold	0.2
Corpus Size	7
Total Word Count	45,152
Corpus Sparsity	99.993%

Influence and Original Factors

Document	Influence	Factors
1	0.21	10
2	0.09	9
3	0.06	3
4	0.06	1
5	0.00	0
6	0.00	0
Author	0.57	127

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.

Influence and Original Factors

Document	Influence	Factors
1	0.21	10
2	0.09	9
3	0.06	3
4	0.06	1
5	0.00	0
6	0.00	0
Author	0.57	127

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.

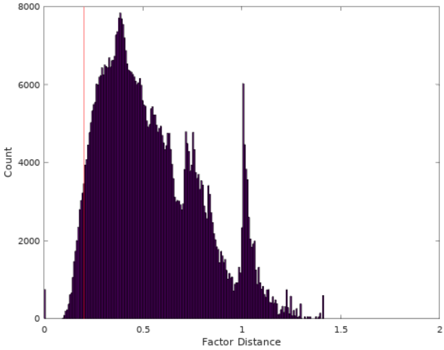
Influence and Original Factors

Document	Influence	Factors
1	0.21	10
2	0.09	9
3	0.06	3
4	0.06	1
5	0.00	0
6	0.00	0
Author	0.57	127

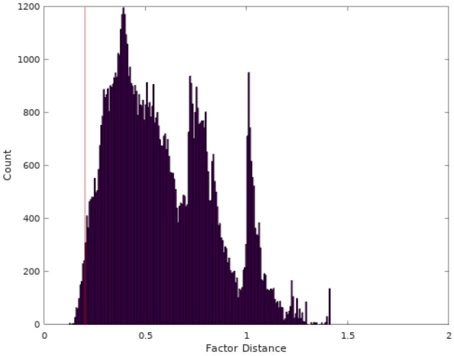
Information From Reading the Target Paper

- 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.

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.

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.

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.

THE UNIVERSITY OF TENNESSEE KNOXVILLE

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

Bibliography I

[1] Alexis Antonia, Hugh Craig, and Jack Elliott.
Language chunking, data sparseness, and the value of a long marker list: explorations with word n-grams and authorial attribution.
Literary and Linguistic Computing, 29(2):147–163, 2014.

[2] John Burrows.
All the way through: testing for authorship in different frequency strata.
Literary and Linguistic Computing, 22(1):27–47, 2006.

Bibliography II

[3] John Burrows and Hugh Craig.
The joker in the pack?: Marlowe, kyd, and the co-authorship of henry vi, part 3.
In Gary Taylor and Gabriel Egan, editors, *The New Oxford Shakespeare Authorship Companion*, chapter 11, pages 194–217. Oxford University Press, 2017.

[4] J. Douglas Carroll and Jih-Jie Chang.
Analysis of individual differences in multidimensional scaling via an n-way generalization of “eckart-young” decomposition.
Psychometrika, 35(3):283–319, Sep 1970.

Bibliography III

[5] Hugh Craig and Arthur F. Kinney.
Sheakespeare, Computers, and the Mystery of Authorship.
Cambridge University Press, 2009.

[6] Richard A Harshman.
Foundations of the parafac procedure: Models and conditions for an" explanatory" multi-modal factor analysis.
1970.

Bibliography IV

[7] Frank L. Hitchcock.
The expression of a tensor or a polyadic as a sum of products.
Journal of Mathematics and Physics, 6(1-4):164–189, 1927.

[8] Noriaki Kawamae.
N-gram over context.
In *Proceedings of the 25th International Conference on World Wide Web*, pages 1045–1055. International World Wide Web Conferences Steering Committee, 2016.

Bibliography V

[9] Henk AL Kiers.
Hierarchical relations among three-way methods.
Psychometrika, 56(3):449–470, 1991.

[10] Tamara G. Kolda and Brett W. Bader.
Tensor decompositions and applications.
SIAM Review, 51(3):455–500, 2009.

[11] Ji Liu, Jun Liu, Peter Wonka, and Jieping Ye.
Sparse non-negative tensor factorization using columnwise coordinate descent.
Pattern Recognition, 45(1):649–656, 2012.