Topic and thesaurus extraction from a document collection

Template *Mathematica* code using NPR transcripts

Anton Antonov
Mathematica for Prediction blog
Mathematica for Prediction project at GitHub
October 2013

Introduction

In this paper we present a template for descriptive statistics analysis and topic and thesaurus extraction for a collection of documents. Both the analysis and topic and thesaurus extraction belong to the field of Natural Language Processing (NLP). The collection of documents used is comprised of National Public Radio (NPR) podcast transcripts, which are available at http://www.npr.org -- see for example http://www.npr.org/templates/transcript/transcript.php?storyId=230950294. (We use nearly 5000 transcripts in this paper.)

The template has the following steps.

1. Ingestion of documents.
2. Removal of stop words and word stemming.
3. Linear vector space representation.
4. Computation of descriptive statistics.
5. Application of different weight functions to the linear vector space representation.
6. Topic extraction with a matrix factorization method.
7. Statistical thesaurus finding using the factorization in step 6.

We describe these steps in detail and give some theoretical clarifications.

For the conversion of documents into points of a linear vector space we use the *Mathematica* package DocumentTermMatrixConstruction.m provided by the project MathematicaForPrediction at GitHub, see [1].

For the topic extraction we use the *Mathematica* package NonNegativeMatrixFactorization.m also provided by the project MathematicaForPrediction at GitHub, see [2].

In general, in this paper we are speak about documents, but we use the word “transcript” when we want to hint the origin of the document.

1. Reading and ingestion of documents

Obviously, the gathering and ingestion of the documents can be done in many ways depending on the sources and storage schemes. With *Mathematica* we can easily ingest from web pages or databases. In any case in this paper we assume that the collection of documents is a list of strings.

Here is a table of the first 100 characters of six randomly selected documents from the collection (which is assigned to the symbol `documents`).

```
In[371]:= Grid[List /@ Map[StringTake[#, {1, 100}] &,
     documents〚RandomInteger[{1, 400}, 6]〛],
  Alignment → Left, Dividers → All]
```

Out[371]=

ROBERT SIEGEL, host:This is ALL THINGS CONSIDERED from NPR News. I'm Robert Siegel.MICHELE NORRIS, h
MELISSA BLOCK, host:Oh, heartbreak.(soundbite of Elvis singing)Elvis Presley's first number one pop
ALEX CHADWICK, host:This DAY TO DAY from NPR News. Readiness for a a possible avian flu epidemic was
RENEE MONTAGNE, host:It's become a holiday tradition on MORNING EDITION to invite commentator M
(Soundbite of music)JENNIFER LUDDEN, host:If I told you this music from a new CD called "H1Bees"--th
ROBERT SIEGEL, host:This is ALL THINGS CONSIDERED from NPR News. I'm Robert Siegel.The artist known

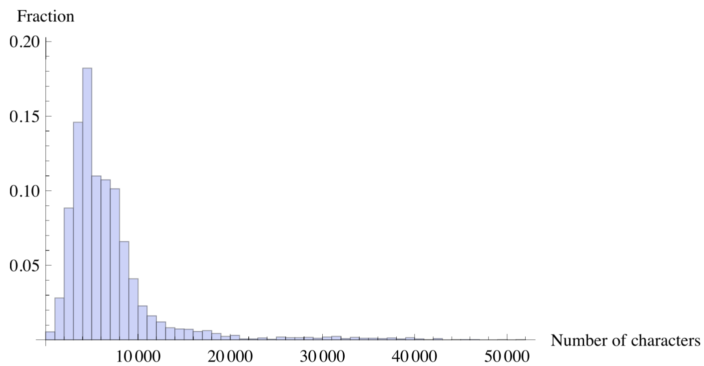
We have $\approx$ 5000 documents:

```
In[415]:= documents // Length
Out[415]= 5123
```

Here is a histogram of their string lengths:

In[414]:=
```
Histogram[StringLength /@ documents, Automatic, "Probability",
 AxesLabel → {"Number of characters", "Fraction"}]
```

Out[414]=



2. Removal of stop words and word stemming

Stop words

In information retrieval “stop words” are removed from texts prior to natural language processing. Loosely speaking stop words have little semantic meaning. See [3].

Here is the list of 319 stop words in English we use (assigned to the symbol `stopWords`):

In[416]:=
```
Magnify[stopWords, 0.7]
```

Out[416]=
```
{a, about, above, across, after, afterwards, again, against, all, almost, alone, along, already,
 also, although, always, am, among, amongst, amoungst, amount, an, and, another, any, anyhow,
 anyone, anything, anyway, anywhere, are, around, as, at, back, be, became, because, become,
 becomes, becoming, been, before, beforehand, behind, being, below, beside, besides, between,
 beyond, bill, both, bottom, but, by, call, can, cannot, cant, co, computer, con, could,
 couldnt, cry, de, describe, detail, do, done, down, due, during, each, eg, eight, either,
 eleven, else, elsewhere, empty, enough, etc, even, ever, every, everyone, everything,
 everywhere, except, few, fifteen, fify, fill, find, fire, first, five, for, former, formerly,
 forty, found, four, from, front, full, further, get, give, go, had, has, hasnt, have, he,
 hence, her, here, hereafter, hereby, herein, hereupon, hers, herself, him, himself, his,
 how, however, hundred, i, ie, if, in, inc, indeed, interest, into, is, it, its, itself,
 keep, last, latter, latterly, least, less, ltd, made, many, may, me, meanwhile, might,
 mill, mine, more, moreover, most, mostly, move, much, must, my, myself, name, namely,
 neither, never, nevertheless, next, nine, no, nobody, none, noone, nor, not, nothing, now,
 nowhere, of, off, often, on, once, one, only, onto, or, other, others, otherwise, our,
 ours, ourselves, out, over, own, part, per, perhaps, please, put, rather, re, same, see,
 seem, seemed, seeming, seems, serious, several, she, should, show, side, since, sincere,
 six, sixty, so, some, somehow, someone, something, sometime, sometimes, somewhere, still,
 such, system, take, ten, than, that, the, their, them, themselves, then, thence, there,
 thereafter, thereby, therefore, therein, thereupon, these, they, thick, thin, third, this,
 those, though, three, through, throughout, thru, thus, to, together, too, top, toward,
 towards, twelve, twenty, two, un, under, until, up, upon, us, very, via, was, we, well, were,
 what, whatever, when, whence, whenever, where, whereafter, whereas, whereby, wherein,
 whereupon, wherever, whether, which, while, whither, who, whoever, whole, whom, whose,
 why, will, with, within, without, would, yet, you, your, yours, yourself, yourselves}
```

Here is a list of additional stop words -- these are words that appear in more than 60% of the NPR transcripts.

Out[554]=

term	%	term	%	term	%
copyright	1.	npr	1.	provided	1.
transcript	1.	host	0.991802	like	0.87156
just	0.865508	soundbite	0.844622	know	0.800703
new	0.776498	time	0.755222	people	0.733555
music	0.726332	news	0.724966	think	0.695881
don	0.686902	really	0.68007	going	0.6748
way	0.670115	years	0.669334	ve	0.654109
called	0.643568	say	0.632247	things	0.623072

The list of additional stop words can be derived with the following commands.

```
In[469]:= wordsTally = Tally[
    Flatten[Map[Complement[Union[Select[StringSplit[ToLowerCase[#],
           {{Whitespace, "\n", "  ", ".", ",", "!", "?", ";",
             ":", "-", "\"", "'", "(", ")", "“", "`"}}],
          StringLength[#] >= 2 &]], stopWords] &, documents]]];
```

```
In[470]:= wordsTally // Length
```

Out[470]= 67 092

```
In[471]:= wordsTally[[1 ;; 45, 1]]
```

Out[471]= {act, ahead, alex, alley, american, apartment, argument, art, ask, assert, attracted, audience, audio, backstage, band, beat, beats, beginning, beginnings, betty, bikini, boring, brings, buns, butter, called, came, carried, cause, chadwick, chance, cinna, cinnamon, computers, copyright, couldn, course, culture, david, day, didn, different, doesn, don, drag}

```
In[472]:= newStopWords =
   SortBy[Select[wordsTally, #[[2]] > 0.6 Length[documents] &], -#[[2]] &];
```

```
In[473]:= newStopWords[[All, 2]] = N[newStopWords[[All, 2]] / Length[documents]];
newStopWords
```

Out[474]= {{copyright, 1.}, {npr, 1.}, {provided, 1.}, {transcript, 1.}, {host, 0.991802}, {like, 0.87156}, {just, 0.865508}, {soundbite, 0.844622}, {know, 0.800703}, {new, 0.776498}, {time, 0.755222}, {people, 0.733555}, {music, 0.726332}, {news, 0.724966}, {think, 0.695881}, {don, 0.686902}, {really, 0.68007}, {going, 0.6748}, {way, 0.670115}, {years, 0.669334}, {ve, 0.654109}, {called, 0.643568}, {say, 0.632247}, {things, 0.623072}, {right, 0.609994}, {got, 0.607261}}

Stemming

Stemming is a process of reducing inflected or derived words to their root, base, or stem;

see [4].

In this paper we are going to use ther word “terms” to mean “stemmed words”.

Here is table with popular terms within the document collection and words that are stemmed to them.

term	words					
abl	able	ables				
creat	create	created	creates	creating		
critic	critic	critical	critically	criticism	criticisms	criticize
die	die	died	dies	dying		
earli	early					
far	far					
month	month	monthly	months			
school	school	schooled	schooling	schools		
second	second	secondly	seconds			
understand	understand	understandable	understandably	understanders	understanding	understandings
walk	walk	walked	walking	walks		

For stemming we can use *Mathematica*’s function `WordData`:

```
In[356]:= WordData[#, "PorterStem"] & /@ {"able", "schooling", "critical"}
Out[356]= {abl, school, critic}
```

Using an external stemmer

We can also use and external stemmer as the stemmer called snowball see http://snowball.-tartarus.org. In this case we do the following steps.

1. Find all individual words used in the document collection.
2. Export all words into a text file.
3. Using the function Run invoke the stemmer with appropriate command arguments.
4. Read the output of the stemmer.
5. Make a list of rules for replacing words with their stems.

Example code using an external stemmer

```
In[475]:= allWords = wordsTally[[All, 1]];

In[476]:= wordsToStem = Complement[Select[allWords,
              StringMatchQ[#, LetterCharacter ..] &], stopWords];
          wordsToStem // Length
Out[477]= 63 241

In[478]:= Export["~/MathFiles/text_words.txt", wordsToStem]
Out[478]= ~/MathFiles/text_words.txt

In[479]:= Run["~/snowball/libstemmer_c/stemwords
            -l english -i ~/MathFiles/text_words.txt
            -o ~/MathFiles/text_words_stemmed.txt"]
Out[479]= 0
```

In[482]:=
```
stemmedWords =
  StringSplit[Import["~/MathFiles/text_words_stemmed.txt"]];
stemmedWords // Length
```

Out[483]= 63 241

In[484]:=
```
stemmingRules = Dispatch[Thread[wordsToStem → stemmedWords]];
```

3. Linear vector space representation

Given a document its words can be taken without regard of their order in the document. We say we turn the document into a “bag of words”. If we use stemming then we turn the document into a bag of terms (stemmed words).

Let us assume that the number of documents in the collection is m and the total number of words used in all documents is n. With the bag-of-words transformation each document can seen as a point in a $\mathbb{R}^n$ linear vector space, each axis of which corresponds to a word. Then the whole document collection can be seen as a sparse matrix in $\mathbb{R}^{m \times n}$.

Assume that we have ordered in some way all the words (terms) in the document collection and in the space of words (terms) $\mathbb{R}^n$ the axis e_w corresponds to the word (term) w. We represent the document D as a point in $\mathbb{R}^n$ in the following way:
1. turn D into a bag of words;
2. stem the words of D;
3. for each term w:
3.1. if w does not appear in D then the coordinate of e_w is 0,
3.2. if w appears f_w times in D then the coordinate of e_w is f_w.

In this representation we can derive the document × term frequency matrix $F \in \mathbb{R}^{m \times n}$ that corresponds to the document collection. The frequency matrix F is further transformed to reflect better the significance of the words in the document collection using different weight functions. (See the section “Weight functions”.)

We can compute the representation of the document collection into a linear vector space with the functions provided in the package DocumentTermMatrixConstruction.m, [1].

In[494]:=
```
Get["~/MathFiles/MathematicaForPrediction/
  DocumentTermMatrixConstruction.m"]
```

The function `DocumentTermMatrix` takes a list of strings and returns a sparse matrix and a list of terms. The returned sparse matrix is the representation of the document collection into a linear vector space with axes corresponding to the returned terms.

```
In[555]:= AbsoluteTiming[
  {F, terms} = DocumentTermMatrix[ToLowerCase /@ documents,
     {stemmingRules, Join[stopWords, newStopWords]}];
 ]
Out[555]= {68.068904, Null}

In[556]:= F
Out[556]= SparseArray[<1 403 565>, {5123, 45 627}]

In[525]:= terms // Length
Out[525]= 45 627
```

Depending on the documents source it can happen that a number of terms are not words or stems of words. For example, in the list of terms found with the previous command using `DocumentTermMatrix` we find more than 3500 terms that are not comprised of letter characters.

```
In[526]:= nonWords = Select[terms, ! StringMatchQ[#, LetterCharacter ..] &];
          nonWords // Length
          RandomSample[nonWords, 12]
Out[527]= 3674
Out[528]= {country…mr, …vaduz, $443, sand…thompson, me…mr, 1925s,
  it…tyler, �and�seabrook, know…gross, 1400s, ‘60s, �abandoned}
```

If we just want to convert a string into a bag of words we can use the function `ToBagOfWords` (which is used by `DocumentTermMatrix`).

```
In[529]:= wordBag = ToBagOfWords[
             ToLowerCase@documents〚1〛, {stemmingRules, stopWords}];
          SortBy[Tally[wordBag], -#〚2〛 &]〚1 ;; 12〛
Out[530]= {{like, 19}, {sing, 16}, {hanna, 15},
  {soundbit, 15}, {song, 13}, {got, 12}, {bikini, 11},
  {kill, 11}, {want, 9}, {band, 8}, {npr, 7}, {tigr, 7}}
```

---

4. Computation of descriptive statistics

Here are some of the basic descriptive statistics we can do over the collection of documents.

1. Total number of documents.
2. Total number of words and total number of stemmed words (terms).
3. Number of terms per document.

4. Number of documents per term.
5. Average number of words in each document.
6. Other statistics, like number of characters, title frequency, etc.

Documents per term

Let us compute descriptive statistics for the number of documents per term.

In[558]:=
```
documentsPerTerm = Total /@ Transpose[Clip[F, {0, 1}]];
TableForm[{{Min, Max, Mean, Median, StandardDeviation},
  Through[{Min, Max, N[Mean[#]] &, Median,
     N[StandardDeviation[#]] &}[documentsPerTerm]]}]
```

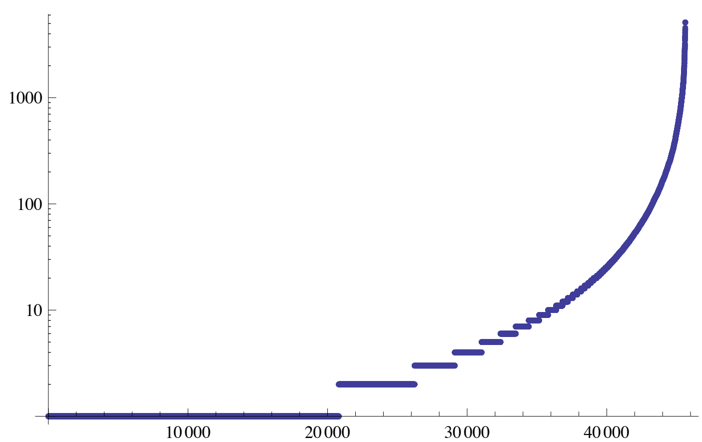
Out[559]//TableForm=

Min	Max	Mean	Median	StandardDeviation
1	5123	30.7617	2	172.999

For this kind of data using `ListLogPlot` is more informative than `Histogram`:

In[560]:=
```
ListLogPlot[Sort[documentsPerTerm], PlotRange → All]
```

Out[560]=



Terms per document

Let us compute descriptive statistics for the number of terms per document.

In[561]:=
```
termsPerDocument = Total /@ Clip[F, {0, 1}];
TableForm[{{Min, Max, Mean, Median, StandardDeviation},
  Through[{Min, Max, N[Mean[#]] &, Median,
     N[StandardDeviation[#]] &}[termsPerDocument]]}]
```

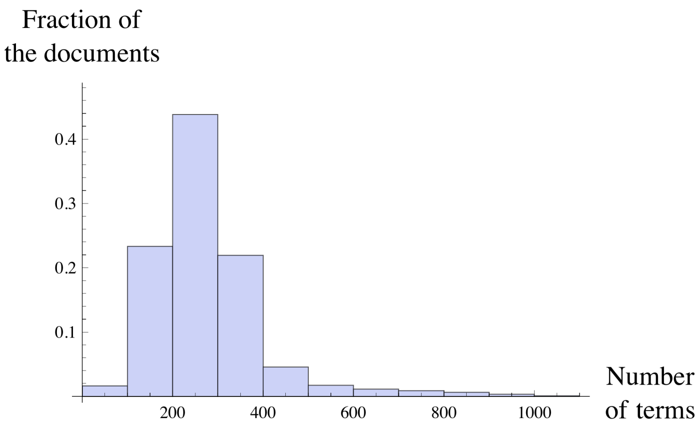
Out[562]//TableForm=

Min	Max	Mean	Median	StandardDeviation
6	1117	273.973	251	125.379

We can get an idea of the terms distribution with a histogram.

In[563]:=
```
Histogram[termsPerDocument, {0, 1100, 100},
  "Probability", AxesLabel → (Style[#, FontSize → 14] & /@
    {"Number\nof terms", "Fraction of\nthe documents"})]
```

Out[563]=



5. Weight functions

We can take the approach used in search engines for calculating weights for document-term matrices. (See [5].)

Frequency matrix

We use the following definitions of the frequency matrix *F*.

Each entry $f_{i\,j}$ of the matrix *F* is the number of occurrences of the term *j* in the list of terms of the document *i*.

Weights

The matrix *F* is transformed into the matrix *M*. Each entry of the matrix *F* is transformed with the formula

$m_{i\,j} = l_{i\,j}\, g_j\, d_i$

where

$l_{i\,j}$ -- local term weight;

g_j -- global term weight;

d_i -- normalization weight.

Various formulas exist for these weights and one of the challenges is to find the right combination for each collection of documents we work with.

Out[550]=

weight type	name	formula
local	Binary	$\chi(f_{i\,j})$
local	Logarithmic	$\log(f_{i\,j}+1)$
local	Term frequency (TF)	$f_{i\,j}$
global	None	1
global	Inverse document frequency (IDF)	$\log\left(\frac{7472}{\sum_j \chi(f_{i\,j})}\right)$
global	Global frequency inverse document frequency (GFIDF)	$\frac{\sum_j f_{i\,j}}{\sum_j \chi(f_{i\,j})}$
global	Normal	$\frac{1}{\sqrt{\sum_i f_{i\,j}^2}}$
normalization	None	1
normalization	Cosine	$\frac{1}{\sqrt{\sum_j g_j\, l_{i\,j}}}$

After applying the chosen weight functions to the elements of *F* we get the matrix *M*. This re-weighting of *F* can be done using the function `WeightTerms` from the package DocumentTermMatrixConstruction.m, [1].

In[601]:=
```
AbsoluteTiming[
 M = WeightTerms[F, GlobalTermWeight["GFIDF", #1, #2] &, # &, # &]
]
```

Out[601]= `{1.403356, SparseArray[<1 403 565>, {5123, 45 627}]}`

6. Topic extraction

Using a matrix factorization method we can extract topics from *M*.

Topic extraction is very similar to dimension reduction and traditionally for dimension reduction the thin Singular Value Decomposition (SVD) is applied to *M*. Because SVD generally produces vectors with mixed positive and negative coordinates we would have difficulties to interpret them into topics.

We use Non-Negative Matrix Factorization (NNMF) for topic extraction from *M*, see [6,7]. The vectors produced by NNMF have positive coordinates and can be easily interpreted. NNMF is not unique (SVD is). NNMF has convergence issues and because of them the initialization of NNMF is important, see [6] for more details.

Describing the algorithms for SVD and NNMF is beyond the scope of this document. Sparse matrix linear algebra libraries usually have SVD implemented. (*Mathematica*'s SVD function is named `SingularValueDecomposition`.)

Topics

Assume we have ten thousand documents, and hence ten thousand bags of words. Topic extraction can be seen as finding a certain number of bags, say 200, for which the following statement is true:

Given a document, 80% of its characterizing words are contained in a small number of the topic bags of words.

We can say that a document is characterized by the topics it consists of. Or in other words the documents are decomposed into topics.

The topics are the rows of the right factor in a SVD or NNMF for the document × term matrix M.

We need to decide which terms comprise a topic. This is best done by some outlier detection procedure. Alternatively, we can simply do the following: given a topic vector t take a certain number of terms that have the largest (and non-zero) coordinates in t.

Theoretical interpretations

Consider the NNMF factorization of $M \in \mathbb{R}^{m\times n}$

$$M \approx W H,\ W \in \mathbb{R}^{m\times k},\ H \in \mathbb{R}^{k\times n},\ W \geq 0,\ H \geq 0. \tag{1}$$

The factorization is derived by solving the (non-linear) optimization problem

$$\begin{gathered}\min \|M - W H\|_F^2,\\ W \geq 0,\\ H \geq 0.\end{gathered} \tag{2}$$

Let us interpret the factors W and H. Each row of the document×term matrix M represents a document in the space of terms. In (1) the integer k is chosen the be much smaller than n, $k <<$ n. The rows of the factor H group the terms into k vectors and those k vectors are used to express each document: each row of H is a basis vector. Assume that (1) is done in such a way that the norms of the rows of H are 1. The i-th row of W, that corresponds to the i-th document in the collection, has coordinates for the basis given by the rows of H. This interpretation follows from the equation

$$M_i \approx \sum_{j=1}^{k} w_{i,j} H_j, \tag{3}$$

in which we denoted with M_i the i-th row of M, with H_j the j-th row of H, and with $w_{i,j}$ the entry of W at row i and column j. We say that each row of H is a topic and with W we have mapped each document into the space of topics. The number of topics is k. In other words with W we reduced the dimension of the document collection matrix representation M.

Using W we can cluster the documents or find nearest neighbors using the Euclidean distance -- if two documents use the same set of topics to a similar degree then these documents are similar.

Note that each column i of W corresponds to a i-th topic (row) in H. Let us denote the i-th column of W with $W(:, i)$. We can reason about the i-th topic properties looking at $W(:, i)$.

If a small fraction of the coordinates of $W(:, i)$ are non-zero and large then that topic is somewhat specialized and does not mesh much with the others. If almost all coordinates of $W(:, i)$ are non-zero then the topic is presented in almost every document and it is probably made of words with little semantic meaning (within the document collection).

Let us take an alternative point of view. We can say that each column of M represents a term in the space of documents in which each document is a basis vector. Assume that we change (1) in such a way that the norms of the columns of W are 1. Then we can cluster the columns of H using the Euclidean distance in oreder to derive a statistical thesaurus based on the document collection.

Note that the basis given by the rows of H is not orthogonal, (2) ensures the positivity of the coordinates of the basis vectors but not their orthogonality.

Computation

In order to extract topics from the document collection we are going to use the NNMF implementation provided by the MathematicaForPrediction project at GitHub, see [2]:

```
In[353]:= Get["~/MathFiles/MathematicaForPrediction/
    NonNegativeMatrixFactorization.m"]
```

First let us select only those terms that are present in at least, say, 15 documents. We can say that the rest of the terms are not significant. We do this mostly to speed up the computations, but also, in effect, we are filtering out terms that do not come from natural language words.

```
In[602]:= pos = Flatten[Position[documentsPerTerm, s_?NumberQ /; s ≥ 15]];
pos // Length

Out[603]= 7744
```

```
In[604]:= M1 = M[[All, pos]]

Out[604]= SparseArray[<1 299 595>, {5123, 7744}]
```

Next we initialize the NNMF factors W and H. The initialization is not necessary since the package function `GDCLS` for computing NNMF does the “standard” initialization of W and H -- the entries of W are random numbers in $[0, 1]$ and all entries of H are 0. The initialization we present here, though, speeds up the convergence and it can be used as a base for more complicated initialization procedures like the ones described in [6]. In order to initialize the i-th column of W we randomly select `p` columns of M and their sum becomes a i-th column of W. (We do this `k` times.) This procedure is done faster if we transpose the matrices M and W.

```
In[628]:= {k, p} = {60, 12};
{m, n} = Dimensions[M1];
M1 = Transpose[M1];
M1 = Map[# &, M1];
H = ConstantArray[0, {k, n}];
W = Table[Total[RandomSample[M1, p]], {k}];
Do[
 W[[i]] = W[[i]] / Norm[W[[i]]];
 , {i, 1, Length[W]}]
W = Transpose[W];
M1 = SparseArray[M1];
M1 = Transpose[M1];
```

The package [2] provides two functions for NNMF: `GDCLS` and `GDCLSGlobal`. The later is used to continue the NNMF factorization iterations for given three symbols associated with the matrices in (1) and hence we can use `GDCLSGlobal` with the initialized factors.

```
In[638]:= W = SparseArray[W];
H = SparseArray[H];
{W, H} = GDCLSGlobal[M1, W, H, "MaxSteps" → 8,
    "PrintProfilingInfo" → True]; // AbsoluteTiming
```

```
1 {202.702771, Null}
2 {208.701122, Null}
3 {214.454326, Null}
4 {210.458800, Null}
5 {211.029350, Null}
6 {213.102886, Null}
7 {212.067769, Null}
8 {213.007171, Null}
Out[640]= {1282.381692, Null}
```

The extracted topics

In order to interpret the rows of H as topics we need to change the product $W H$ in such a way that the norms of the rows of H are 1. This can be done with the function `RightNormalizeMatrixProduct` of [2]:

```
In[646]:= {W, H} = RightNormalizeMatrixProduct[W, H];
```

In order to print out the interpretations of the rows of H as topics we need to convert H from a sparse array to a list of lists structure. (We do this for W too.)

```
In[647]:= {W, H} = Normal /@ {W, H};
```

The function `BasisVectorInterpretation` of [2] can be used to get the larges coordinates of a vector and find the terms corresponding to them.

In[749]:=
```
BasisVectorInterpretation[H[[2]], 12, terms[[pos]]]
```

Out[749]=
```
{{105.967, gross}, {63.6485, music}, {50.8828, record},
 {41.5973, play}, {40.4996, band}, {35.5987, song},
 {33.7273, sing}, {29.3061, cash}, {28.2842, did},
 {26.5285, soundbit}, {26.4699, album}, {24.5975, jone}}
```

Now we can construct a table of topics.

In[649]:=
```
topicsTbl =
  Table[
   (
    t = BasisVectorInterpretation[H[[ind]], 12, terms[[pos]]];
    TableForm[{NumberForm[#[[1]] / t[[1, 1]], {4, 3}], #[[2]]} & /@ t]
   ), {ind, 1, k}];
```

In[750]:=
```
Magnify[#, 0.68] &@Grid[Partition[
    ColumnForm /@ Transpose[{Style[#, Red] & /@ Range[k], topicsTbl}],
    5], Dividers → All, Alignment → Left]
```

1		2		3		4		5	
1.000	song	1.000	gross	1.000	know	1.000	sing	1.000	gross
0.461	gross	0.601	music	0.945	gross	0.528	watson	0.868	sondheim
0.430	write	0.480	record	0.596	like	0.475	hard	0.315	song
0.353	word	0.393	play	0.338	just	0.332	know	0.282	music
0.332	lyric	0.382	band	0.208	jay	0.297	guitar	0.276	levin
0.229	wrote	0.336	song	0.184	did	0.270	don	0.268	thing
0.211	jay	0.318	sing	0.148	yeah	0.243	good	0.224	chord
0.211	day	0.277	cash	0.134	russel	0.215	streisand	0.210	want
0.199	sondheim	0.267	did	0.129	mother	0.212	realli	0.200	write
0.193	like	0.250	soundbit	0.128	stew	0.201	applaus	0.191	rhyme
0.180	frank	0.250	album	0.123	new	0.199	come	0.180	piano
0.177	peopl	0.232	jone	0.120	cash	0.183	littl	0.169	did
6		7		8		9		10	
1.000	sing	1.000	martin	1.000	stewart	1.000	gross	1.000	mcdonald
0.939	inskeep	0.193	just	0.996	just	0.868	lynn	0.817	bess
0.923	npr	0.157	peopl	0.983	record	0.602	song	0.577	gross
0.629	morn	0.120	like	0.737	langer	0.512	old	0.501	porgi
0.620	montagn	0.105	michel	0.715	dog	0.493	know	0.441	sing
0.568	soundbit	0.104	go	0.680	soundbit	0.364	got	0.367	know
0.556	song	0.097	say	0.605	like	0.349	low	0.311	just
0.462	green	0.092	want	0.604	make	0.344	soundbit	0.253	time
0.402	say	0.090	don	0.570	chideya	0.321	record	0.231	opera
0.390	hard	0.084	thank	0.552	got	0.300	laughter	0.208	crown
0.371	host	0.080	yeah	0.499	tour	0.266	yeah	0.202	like
0.355	year	0.079	npr	0.497	work	0.264	play	0.177	soundbit

11		12		13		14		15	
1.000	conan	1.000	johnson	1.000	know	1.000	say	1.000	like
0.211	thank	0.808	play	0.826	think	0.919	npr	0.417	know
0.195	talk	0.569	said	0.209	realli	0.885	peopl	0.401	gross
0.134	know	0.473	like	0.203	peopl	0.805	year	0.343	metal
0.121	nation	0.402	sit	0.197	don	0.788	money	0.279	band
0.116	yeah	0.366	kennedi	0.180	thing	0.696	work	0.270	heavi
0.114	peopl	0.355	say	0.179	want	0.663	make	0.248	just
0.114	go	0.349	hard	0.175	just	0.625	like	0.199	music
0.105	new	0.342	just	0.155	mean	0.530	dollar	0.148	baghdad
0.105	yes	0.338	time	0.147	talk	0.430	jaff	0.144	live
0.101	neal	0.313	man	0.129	sort	0.373	movi	0.142	iraq
0.099	caller	0.308	soundbit	0.124	go	0.369	thing	0.140	yeah

16		17		18		19		20	
1.000	patient	1.000	song	1.000	music	1.000	know	1.000	jone
0.468	doctor	0.639	sing	0.434	peopl	0.328	yeah	0.629	jame
0.407	care	0.561	countri	0.414	women	0.326	stewart	0.360	gross
0.391	use	0.376	bon	0.391	say	0.317	go	0.288	record
0.372	dorsey	0.375	nelson	0.340	think	0.275	record	0.260	roth
0.253	say	0.349	nashvill	0.318	npr	0.274	right	0.251	know
0.252	provid	0.339	record	0.315	movement	0.272	martin	0.247	sharon
0.245	just	0.294	just	0.289	realli	0.255	langer	0.240	just
0.233	medic	0.285	like	0.280	kind	0.244	good	0.206	like
0.230	health	0.258	soundbit	0.259	soundbit	0.244	food	0.194	band
0.214	npr	0.224	doe	0.254	american	0.224	minut	0.163	soundbit
0.209	john	0.215	new	0.216	year	0.208	say	0.154	look

21		22		23		24		25	
1.000	know	1.000	campaign	1.000	carney	1.000	peopl	1.000	monk
0.612	hansen	0.885	obama	0.538	like	0.963	children	0.709	gross
0.560	eisenberg	0.850	mccain	0.444	know	0.845	said	0.586	like
0.422	laughter	0.829	peopl	0.363	gross	0.840	child	0.548	kelley
0.371	yeah	0.589	think	0.357	just	0.838	say	0.437	just
0.365	actual	0.530	like	0.354	black	0.726	year	0.247	theloni
0.260	right	0.483	say	0.335	key	0.636	crime	0.239	wild
0.227	play	0.457	sort	0.328	record	0.580	npr	0.231	way
0.214	mean	0.456	just	0.325	yeah	0.564	rape	0.224	peopl
0.209	just	0.429	candid	0.302	song	0.540	case	0.222	think
0.184	soundbit	0.350	talk	0.302	realli	0.510	gun	0.217	soundbit
0.158	sagal	0.323	barack	0.280	soundbit	0.484	law	0.210	love

26		27		28		29		30	
1.000	life	1.000	senat	1.000	black	1.000	say	1.000	nuclear
0.630	know	0.964	republican	0.811	reed	0.573	gun	0.878	plant
0.530	song	0.947	obama	0.634	peopl	0.558	militari	0.831	davi
0.443	music	0.917	elect	0.616	race	0.539	peopl	0.773	edg
0.321	soundbit	0.896	democrat	0.551	white	0.529	block	0.657	worker
0.231	time	0.878	presid	0.492	biraci	0.484	zwerdl	0.485	radiat
0.217	year	0.772	vote	0.446	american	0.466	npr	0.448	tsunami
0.196	abort	0.768	state	0.376	prof	0.421	year	0.434	power
0.194	thing	0.642	think	0.365	doe	0.393	presid	0.403	fuel
0.183	day	0.583	go	0.365	just	0.375	know	0.370	happen
0.179	realli	0.549	parti	0.350	identifi	0.374	go	0.349	meltdown
0.179	say	0.511	mccain	0.346	parent	0.348	right	0.325	japan

Out[750]=

31		32		33		34		35	
1.000	wait	1.000	ari	1.000	book	1.000	song	1.000	gordon
0.506	gross	0.310	like	0.594	armstrong	0.960	sing	0.693	know
0.419	like	0.256	just	0.470	right	0.813	simon	0.281	yeah
0.317	yeah	0.249	thank	0.463	polit	0.580	soundbit	0.269	sing
0.305	know	0.248	conan	0.387	neari	0.377	streisand	0.245	peopl
0.301	song	0.237	sing	0.382	martin	0.336	laughter	0.234	music
0.163	don	0.228	soundbit	0.345	inskeep	0.198	thank	0.195	just
0.151	soundbit	0.194	india	0.333	npr	0.196	burnett	0.159	talk
0.151	tom	0.187	laughter	0.309	yeah	0.189	gross	0.158	like
0.145	want	0.180	talk	0.298	go	0.183	did	0.149	want
0.142	sing	0.147	chorus	0.268	read	0.169	think	0.141	right
0.136	new	0.141	hair	0.263	chideya	0.161	favorit	0.141	say

36		37		38		39		40	
1.000	song	1.000	david	1.000	song	1.000	raz	1.000	flatow
0.738	soundbit	0.914	gross	0.659	sing	0.476	song	0.336	scienc
0.390	block	0.834	promis	0.647	love	0.424	guy	0.266	yeah
0.371	sing	0.733	bacharach	0.334	soundbit	0.415	like	0.202	peopl
0.357	road	0.298	did	0.202	just	0.296	band	0.157	go
0.262	play	0.289	record	0.195	album	0.292	record	0.151	come
0.200	guitar	0.281	burt	0.144	block	0.269	soundbit	0.143	right
0.198	band	0.270	hal	0.125	singer	0.238	know	0.125	make
0.191	just	0.251	yeah	0.125	new	0.228	yeah	0.116	say
0.191	laughter	0.244	write	0.112	record	0.188	play	0.111	don
0.191	yeah	0.242	fall	0.110	got	0.183	call	0.109	like
0.187	cash	0.241	time	0.106	like	0.163	just	0.109	element

41		42		43		44		45	
1.000	music	1.000	like	1.000	languag	1.000	music	1.000	unidentifi
0.937	new	0.427	say	0.920	foreign	0.409	soundbit	0.586	man
0.498	play	0.361	day	0.760	music	0.287	compos	0.506	soundbit
0.484	jazz	0.355	thing	0.756	sing	0.265	piec	0.505	sing
0.455	york	0.311	wear	0.517	soundbit	0.194	huizenga	0.331	music
0.335	soundbit	0.283	look	0.295	npr	0.167	classic	0.300	npr
0.292	record	0.279	right	0.260	song	0.154	npr	0.265	woman
0.276	musician	0.252	hard	0.257	spoken	0.147	symphoni	0.252	say
0.263	orlean	0.247	just	0.249	singer	0.134	new	0.249	like
0.224	npr	0.246	npr	0.240	say	0.119	orchestra	0.206	group
0.184	citi	0.237	don	0.185	african	0.117	think	0.193	year
0.178	like	0.229	make	0.166	opera	0.105	hear	0.178	peopl

46		47		48		49		50	
1.000	simon	1.000	smith	1.000	ray	1.000	sing	1.000	play
0.237	hansen	0.179	just	0.474	sing	0.788	sweet	0.508	music
0.201	soundbit	0.166	like	0.329	just	0.781	honey	0.342	soundbit
0.175	song	0.127	robert	0.274	yeah	0.767	rock	0.303	sound
0.170	yeah	0.085	npr	0.200	ami	0.388	children	0.271	like
0.143	music	0.081	new	0.193	girl	0.359	gonna	0.257	instrument
0.124	just	0.073	patti	0.161	like	0.303	say	0.256	organ
0.118	scott	0.071	know	0.152	dave	0.287	martin	0.255	just
0.109	thank	0.071	littl	0.146	emili	0.276	spirit	0.247	guitar
0.107	npr	0.070	martin	0.146	activ	0.228	think	0.221	record
0.093	read	0.068	soundbit	0.129	sondheim	0.211	soundbit	0.216	gross
0.092	like	0.068	say	0.111	conan	0.189	stranger	0.209	hansen

51		52		53		54		55	
1.000	simon	1.000	deal	1.000	like	1.000	jazz	1.000	sagal
0.969	mar	0.751	pesca	0.792	kind	0.970	record	0.501	laughter
0.455	earth	0.650	kim	0.705	know	0.674	blue	0.381	like
0.425	moon	0.508	kelley	0.466	just	0.611	musician	0.343	just
0.391	elvi	0.454	song	0.435	song	0.545	music	0.222	lewi
0.355	space	0.267	did	0.380	realli	0.465	note	0.195	say
0.337	peopl	0.212	come	0.256	yeah	0.432	soundbit	0.170	play
0.324	soundbit	0.200	home	0.225	music	0.408	brian	0.169	did
0.266	come	0.171	realli	0.220	album	0.407	busi	0.158	said
0.265	go	0.164	play	0.186	sort	0.394	say	0.153	time
0.246	way	0.162	folk	0.178	mean	0.393	npr	0.152	think
0.229	know	0.158	record	0.175	band	0.334	make	0.150	right
56		57		58		59		60	
1.000	woodi	1.000	music	1.000	gross	1.000	pesca	1.000	conan
0.735	guthri	0.663	soundbit	0.486	day	0.901	leo	0.909	blue
0.594	song	0.275	play	0.400	know	0.615	yeah	0.861	note
0.486	know	0.211	like	0.309	brain	0.555	like	0.589	record
0.367	place	0.209	npr	0.307	rain	0.399	laughter	0.275	thank
0.313	dust	0.202	song	0.243	lincoln	0.397	right	0.260	just
0.273	land	0.182	block	0.206	did	0.353	hard	0.239	jone
0.248	jeff	0.164	band	0.192	record	0.303	read	0.233	year
0.224	sing	0.156	sound	0.187	imag	0.291	simon	0.221	album
0.220	got	0.148	jazz	0.177	low	0.280	know	0.218	said
0.211	peopl	0.135	sing	0.166	work	0.257	applaus	0.214	jazz
0.208	set	0.129	album	0.163	terri	0.242	play	0.202	time

7. Statistical thesaurus

We can also find a statistical thesaurus that fits the body of the documents. For example, the words “pollution”, “fossil”, ”greenhouse”, “gasoline” are found together in the NPR transcripts.

The statistical thesaurus for the *i*-th term can be found by taking, say, 20 nearest neighbors of the *i*-th column from the right matrix factor in a SVD or NNMF (using the Euclidean distance).

Computation

In order to find a statistical thesaurus for the collection of documents represented with M we normalize the product $W\,H$ in such a way that the norms of the columns of W are 1. (The alternative normalization making the norms of the rows of H to be 1 uses a different point of view of what is a statistical thesaurus.)

In[663]:=
```
{W, H} = NormalizeMatrixProduct[W, H];
```

Instead using clustering we are going to demonstrate the thesaurus finding using nearest neighbors. So, we pre-compute the following nearest neighbors function:

```
In[664]:= HNF = Nearest[Range[Dimensions[H]〚2〛],
            DistanceFunction → (Norm[H〚All, #1〛 - H〚All, #2〛] &)]

Out[664]= NearestFunction[{7744, 1}, <>]
```

Next we define a function that would find the thesaurus entry for a given word:

```
In[665]:= Clear[StatThesaurus];
          StatThesaurus[word_String, n_Integer: 20] :=
            Block[{sword, tpos, inds},
             sword = word /. stemmingRules;
             tpos = Position[terms〚pos〛, sword];
             If[Length[tpos] == 0, {},
              inds = HNF[tpos〚1, 1〛, n];
              terms〚pos〛〚inds〛
             ]
            ];
```

Here is a table of invoking `StatThesaurus` over a set of words:

```
In[671]:= Magnify[#, 0.7] &@
           Grid[Prepend[Map[{#, StatThesaurus[#, 15]} &, {"senate", "obama",
                "war", "food", "fbi", "singer", "jazz", "school", "homeland"}],
             Style[#, Blue, FontFamily → "Times"] & /@
              {"word", "statistical thesaurus"}], Dividers → All,
            Alignment → Left, Spacings → {Automatic, 0.75}]
```

Out[671]=

word	statistical thesaurus
senate	{senat, elect, republican, democrat, vote, presid, parti, state, ken, polit, voter, poll, governor, obama, barack}
obama	{obama, mccain, campaign, senat, elect, vote, polit, presid, republican, barack, state, democrat, parti, race, candid}
war	{war, member, program, unit, iraq, forc, number, command, job, general, ground, washington, refuge, situat, soldier}
food	{food, eat, minut, fessler, grow, buy, garden, usual, hour, veget, basic, tomato, ann, stamp, hunger}
fbi	{fbi, suspici, enforc, prosecutor, juri, prosecut, sentenc, surveil, brutal, district, incid, bryan, punish, violat, chapter}
singer	{singer, voic, babi, heart, beauti, soul, heard, gospel, roll, produc, dream, norri, away, listen, track}
jazz	{jazz, musician, york, artist, orlean, label, citi, rose, piano, tune, heard, art, pianist, busi, player}
school	{school, help, take, ago, watch, stop, job, student, men, took, everybodi, friend, head, number, get}
homeland	{homeland, blade, enhanc, manipul, dispos, conscienc, poorer, correl, medit, brake, phase, invad, psychiatrist, stimul, lash}

References

[1] Anton Antonov., Implementation of document-term matrix construction and re-weighting functions in *Mathematica*, source code at GitHub, https://github.com/antononcube/MathematicaForPrediction, package DocumentTermMatrixConstruction.m, (2013).

[2] Anton Antonov, Implementation of non-negative matrix factorization in *Mathematica*, source code at GutHub, https://github.com/antononcube/MathematicaForPrediction, package NonNegativeMatrixFactorization.m, (2013).

[3] Stop words, Wikipedia entry, http://en.wikipedia.org/wiki/Stop_words .

[4] Stemming, Wikipedia entry, http://en.wikipedia.org/wiki/Stemming .

[5] Michael Berry, Murray Browne, "Understanding Search Engines: Mathematical Modeling and Text Retrieval". SIAM, 2005.
http://books.google.com/books/about/Understanding_Search_Engines.html?id=J21ooXWVdzkC
http://www.amazon.com/Understanding-Search-Engines-Mathematical-Environments/dp/0898715814

[6] Russell Albright, et al., Algorithms, Initializations, and Convergence for the Nonnegative Matrix Factorization, http://meyer.math.ncsu.edu/meyer/ps_files/nmfinitalgconv.pdf

[7] Michael Berry, et al., Algorithms and Applications for Approximate Nonnegative Matrix Factorization, preprint Elsevier Preprint (2006), http://users.wfu.edu/plemmons/papers/BBLPP-rev.pdf.