CS4487 - Machine Learning

Lecture 7 - Linear Dimensionality Reduction

Dr. Antoni B. Chan

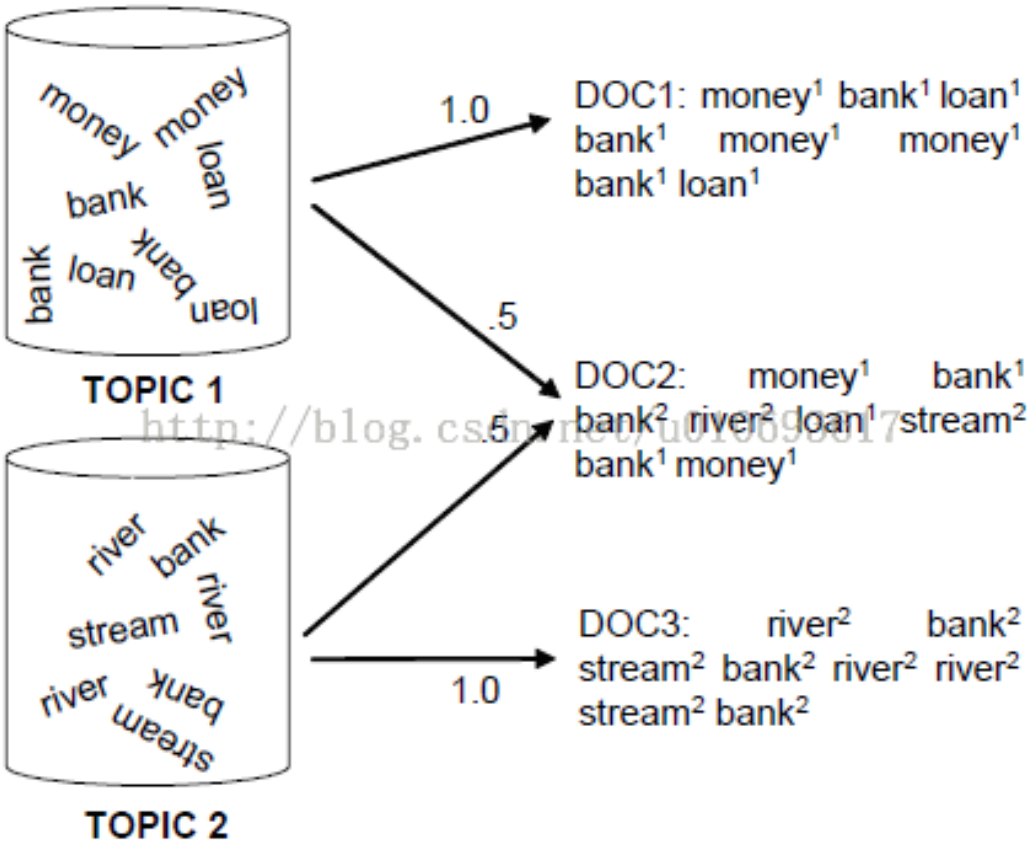
Dept. of Computer Science, City University of Hong Kong

Outline

1. Linear Dimensionality Reduction for Vectors
   A. Principal Component Analysis (PCA)
   B. Random Projections
   C. Fisher's Linear Discriminant (FLD)
2. **Linear Dimensionality Reduction for Text**
   A. Latent Semantic Analysis (LSA)
   B. Non-negative Matrix Factorization (NMF)
   C. Latent Dirichlet Allocation (LDA)

Dimensionality Reduction

- **Goal:** Transform high-dimensional vectors into low-dimensional vectors.
  - Dimensions in the low-dim data represent co-occuring features in high-dim data.
  - Dimensions in the low-dim data may have semantic meaning.
- **For example:** document analysis
  - high-dim: bag-of-word vectors of documents
  - low-dim: each dimension represents similarity to a topic.



Latent Semantic Analysis (LSA)

- Also called *Latent Semantic Indexing*
- Consider a bag-of-word representation (e.g., TF, TF-IDF)
  - document vector $\mathbf{x}_i$
  - $x_{i,j}$ is the frequency of word j in document i

- Approximate each document vector as a weighted sum of topic vectors.
  - $\hat{\mathbf{x}} = \sum_{n=1}^{p} w_p \mathbf{v}_p$
  - Topic vector $\mathbf{v}_p$ contains co-occuring words.
    - corresponds to a particular *topic* or *theme*.
  - Weight w_p represents similarity of the document to the p-th topic.
- Objective:
  - minimize the squared reconstruction error (Similar to PCA):
  - $\min_{\mathbf{v},\mathbf{w}} \sum_i ||\mathbf{x}_i - \hat{\mathbf{x}_i}||^2$

- Represent each document by its topic weights.
  - Apply other machine learning algorithms...
- **Advantage:**
  - Finds relations between terms (synonymy and polysemy).
  - distances/similarities are now comparing topics rather than words.
    - higher-level semantic representation

Example on Spam Email dataset

- use bag-of-words representation with 50 words
- term-frequency (TF) normalization

In [2]:
```
# Load spam/ham text data from directories
textdata = datasets.load_files("email", encoding="utf8", decode_error="replace")

# convert to bag-of-words representation
cntvect = feature_extraction.text.CountVectorizer(stop_words='english', max_featur
es=50)
X = cntvect.fit_transform(textdata.data)
Y = textdata.target

# TF representation
tf_trans = feature_extraction.text.TfidfTransformer(norm='l1', use_idf=False)
Xtf = tf_trans.fit_transform(X)

# print the vocabulary
print(cntvect.vocabulary_)
```

```
{'10': 2, 'brands': 13, 'office': 38, 'isidoro': 29, 'contact': 17, 'john': 30
, 'good': 23, 'today': 46, 'files': 22, '15mg': 4, 'email': 21, '120': 3, 'goo
gle': 24, 'payment': 40, 'answer': 10, 'inform': 27, 'percocet': 41, '30mg': 6
, 'codeine': 15, 'peter': 42, 'mr': 34, 'just': 31, 'address': 9, '30': 5, '50
': 7, 'watches': 49, 'number': 37, 'com': 16, 'buy': 14, 'bags': 11, 'bank': 1
2, 'new': 35, '000': 1, 'online': 39, 'cost': 18, 'mg': 33, 'visa': 48, 'pills
': 43, 'united': 47, 'hi': 25, 'status': 44, 'store': 45, '60': 8, '00': 0, 'i
nformation': 28, 'country': 19, 'http': 26, 'day': 20, 'nigeria': 36, 'kamara'
: 32}
```

LSA on Spam data

- Apply LSA with 5 topics
  - implemented as `TruncatedSVD`

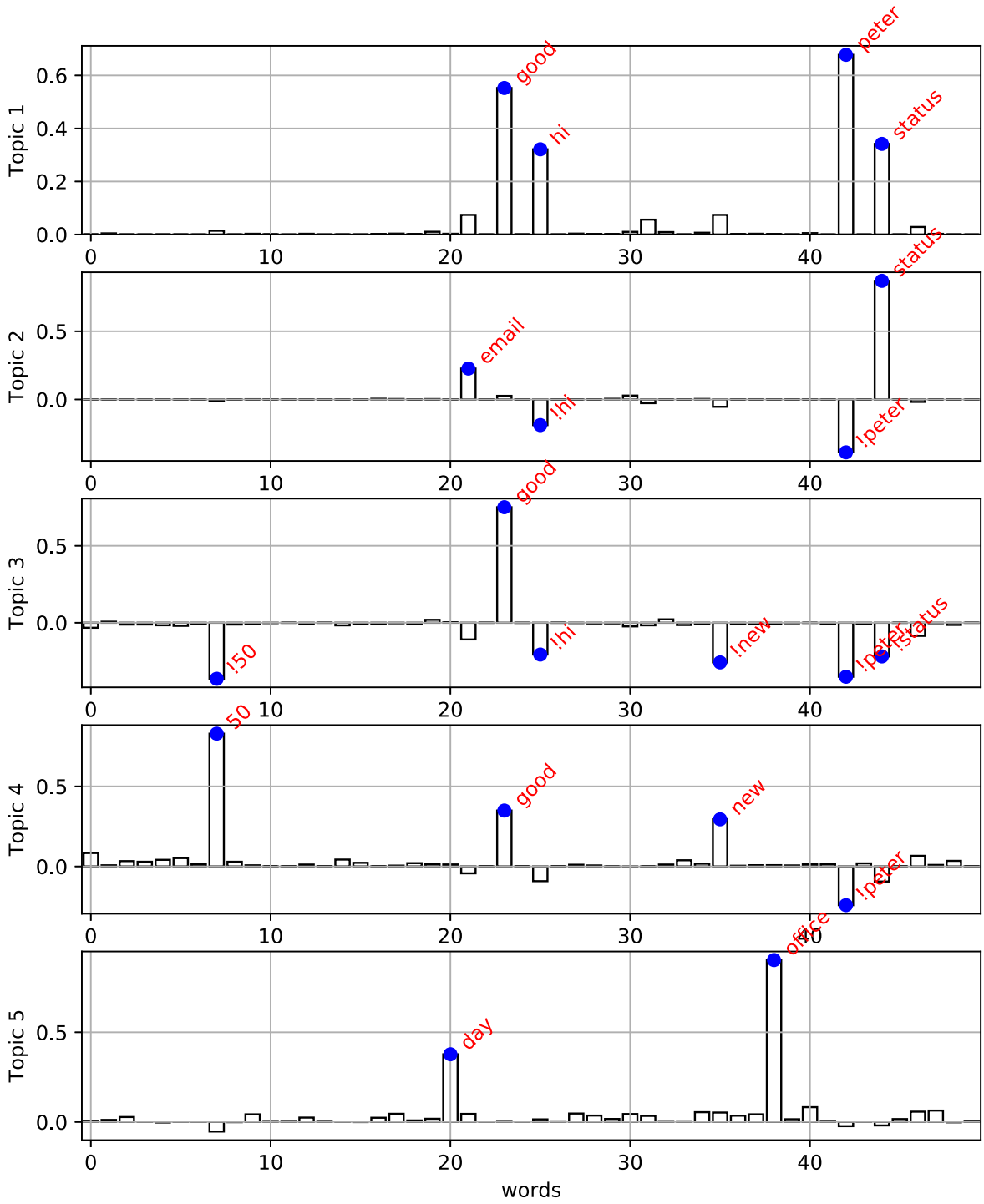
In [3]:
```
lsa = decomposition.TruncatedSVD(n_components=5, random_state=4487)
Wlsa = lsa.fit_transform(Xtf)

# components
V = lsa.components_
```

Topic vectors

- topic vectors contain frequent co-occuring words

In [5]:
```
vocab = asarray(cntvect.get_feature_names())
lsafig = plot_topics(lsa, vocab)
lsafig
```

Out[5]:

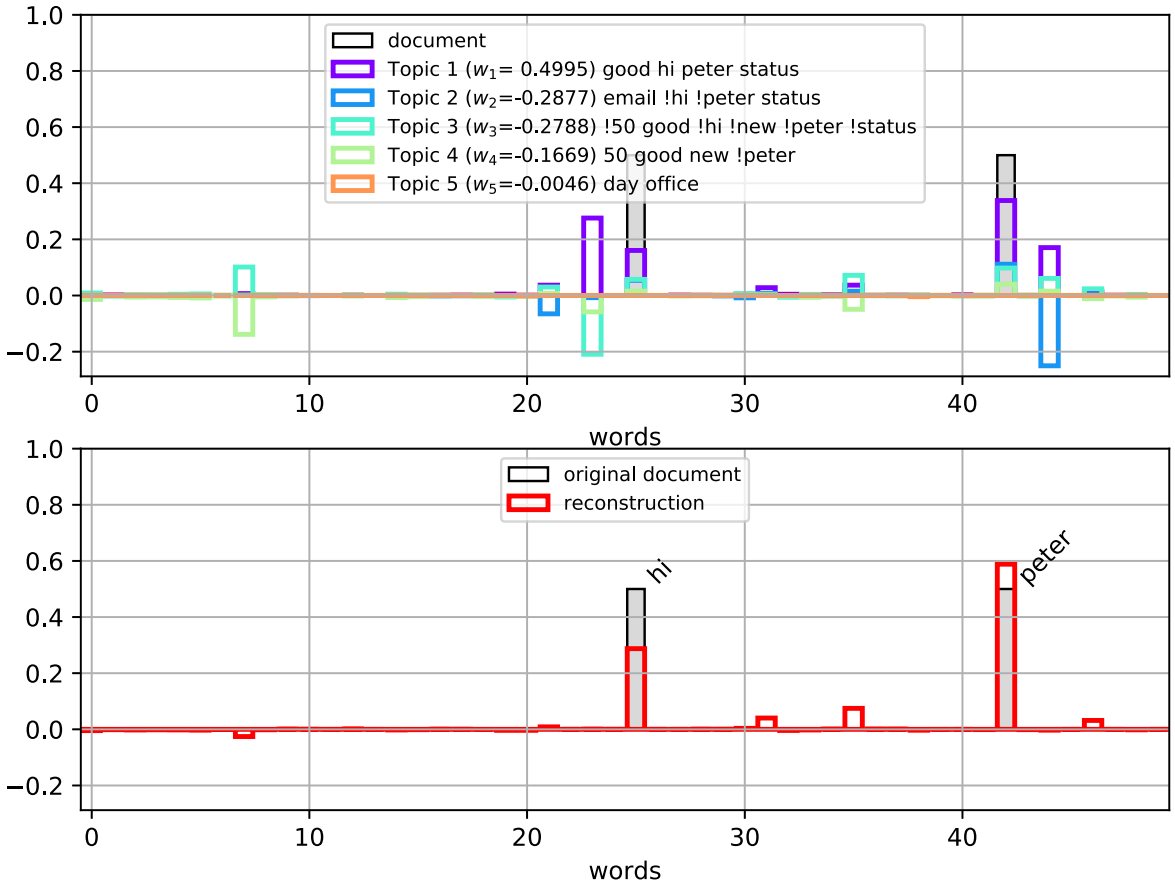


Document representation

- Documents are a combination of topics

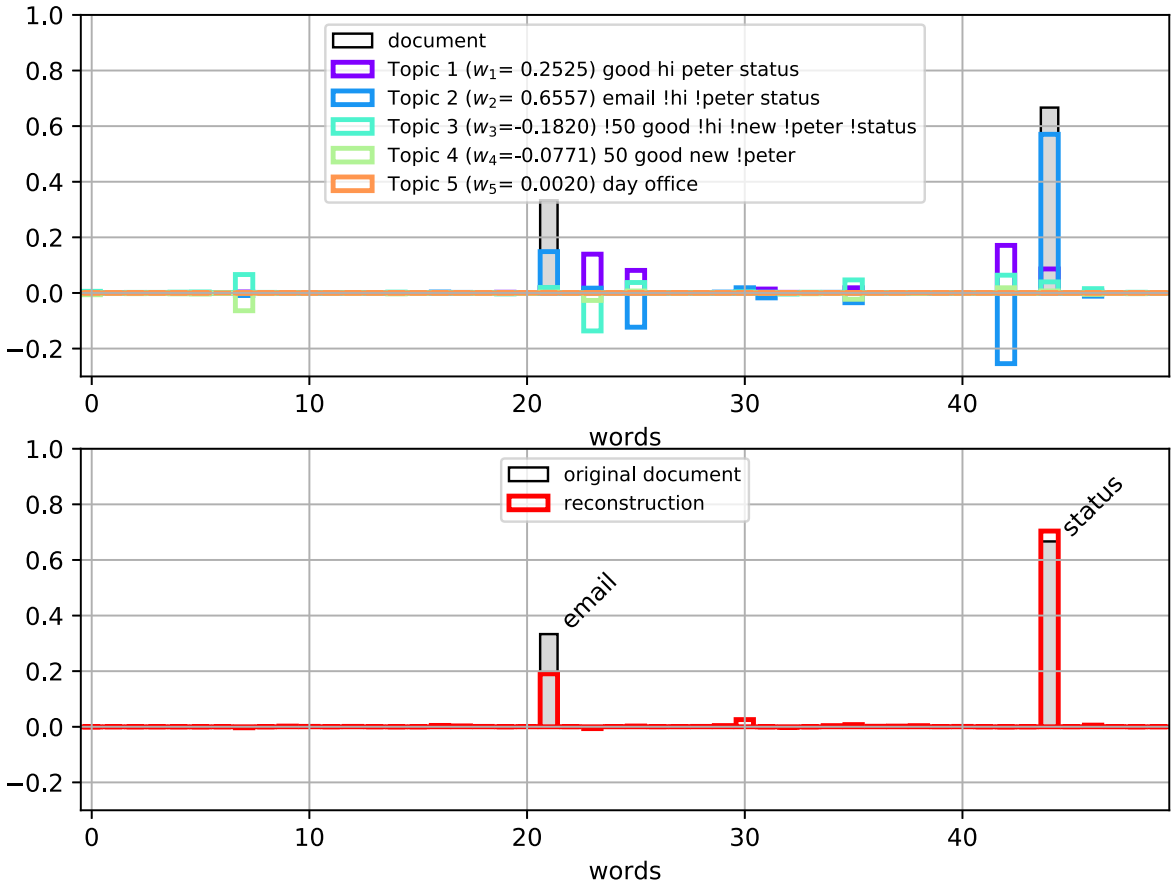
In [7]:
```
plot_doc_topic(Xtf[2,:], Wlsa[2,:], lsa, vocab)
```

Out[7]:



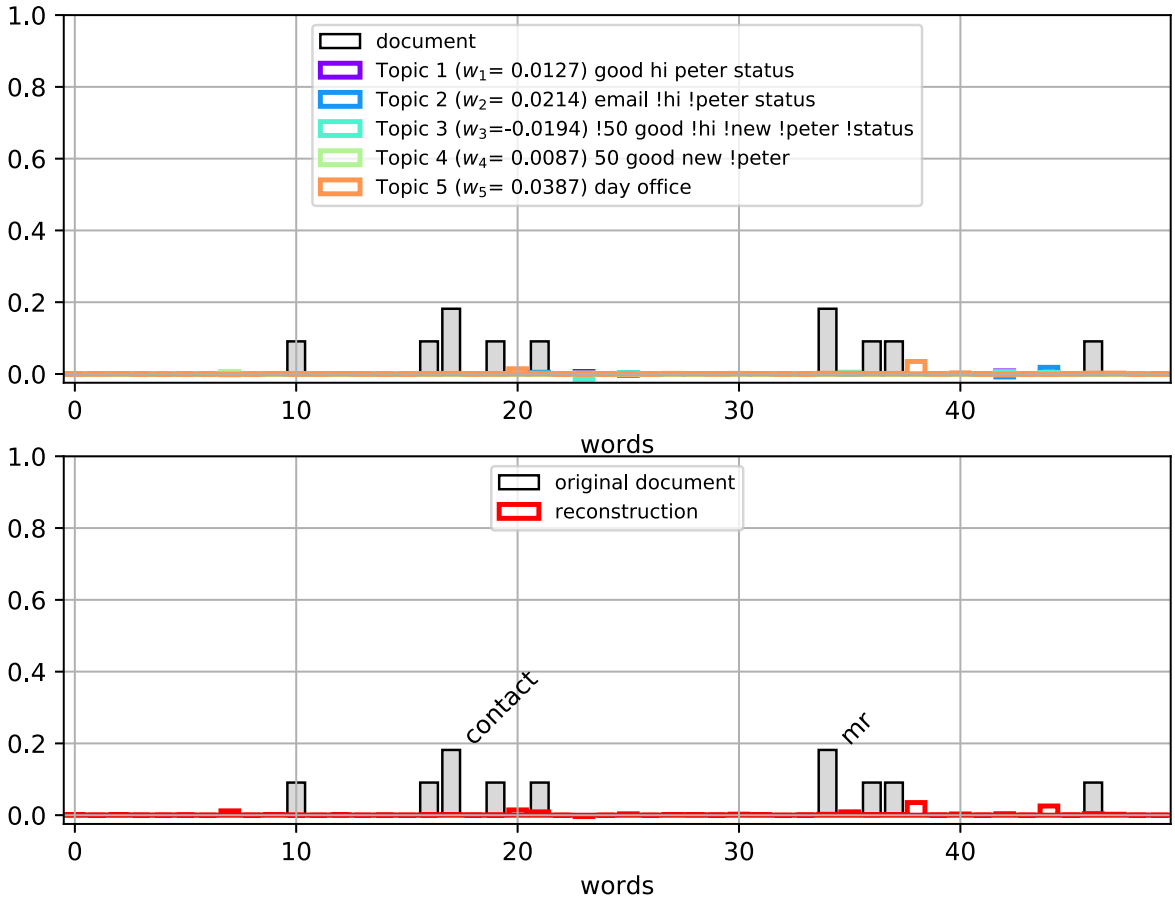
In [8]:
```
plot_doc_topic(Xtf[4,:], Wlsa[4,:], lsa, vocab)
```

Out[8]:



In [9]:
```
plot_doc_topic(Xtf[3,:], Wlsa[3,:], lsa, vocab)
```

Out[9]:

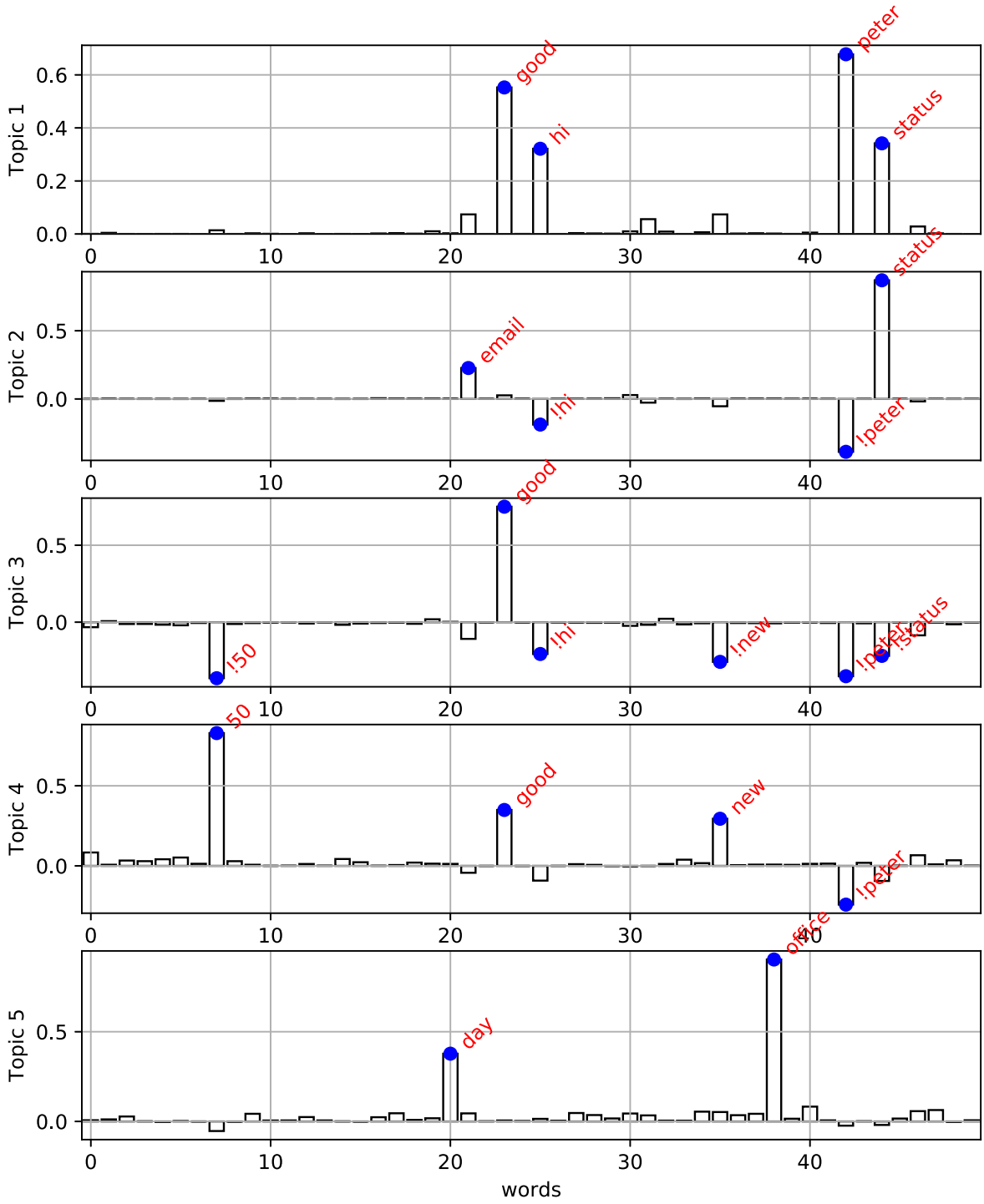


Problem with LSA

- In the topic vector, the "frequency" of a word can be negative!
  - Doesn't really make sense for document bag-of-words model.

In [10]:
```
lsafig
```

Out[10]:

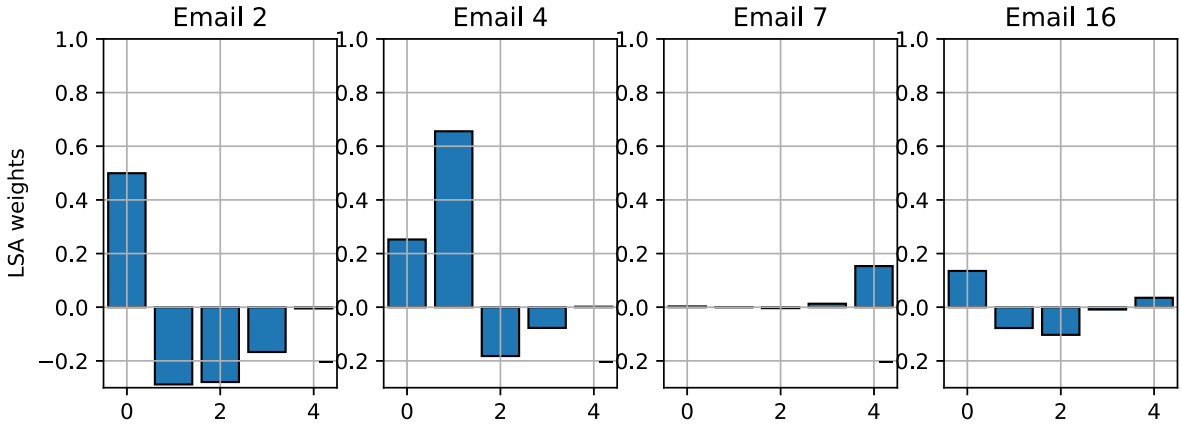


Problems with LSA

- The weights for each topic can be negative!
  - Topics should only be "additive"
    - Topics should increase probability of some topic-related words, but not decrease probability of other words.
    - It doesn't make sense to "remove" a topic using a negative topic weight.

In [12]:
```
pfig
```

Out[12]:



Non-negative Matrix Factorization (NMF)

- **Solution:** constrain the topic vector and weights to be non-negative.
- Similar to LSA
  - Approximate each document vector as a weighted sum of topic vectors.
    - $\hat{\mathbf{x}_j} = \sum_{n=1}^{p} w_p \mathbf{v}_p$
    - But now, each entry of topic vector $\mathbf{v}_p \geq 0$ and topic weight $w_p \geq 0$
  - Objective: minimize the squared reconstruction error
    - $\min_{\mathbf{v},\mathbf{w}} \sum_j ||\mathbf{x}_j - \hat{\mathbf{x}_j}||^2$
    - subject to the non-negative constraints.

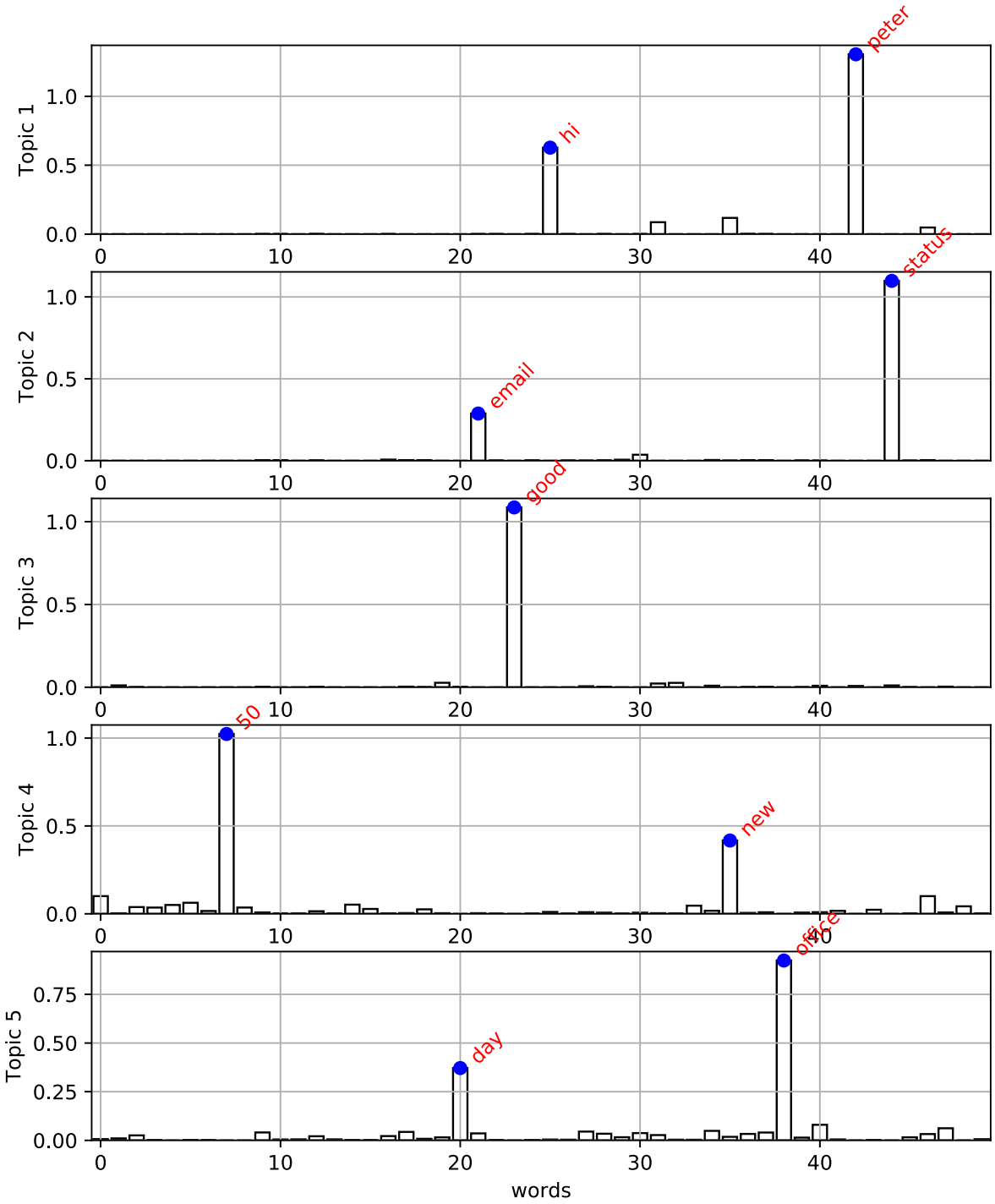
In [13]:
```
# Run NMF
nmf = decomposition.NMF(n_components=5)
Wnmf = nmf.fit_transform(Xtf)

# components
V = nmf.components_
```

Topic vector

- all non-negative entries
- looks much cleaner (less small entries)

In [14]:
```
plot_topics(nmf, vocab)
```

Out[14]:

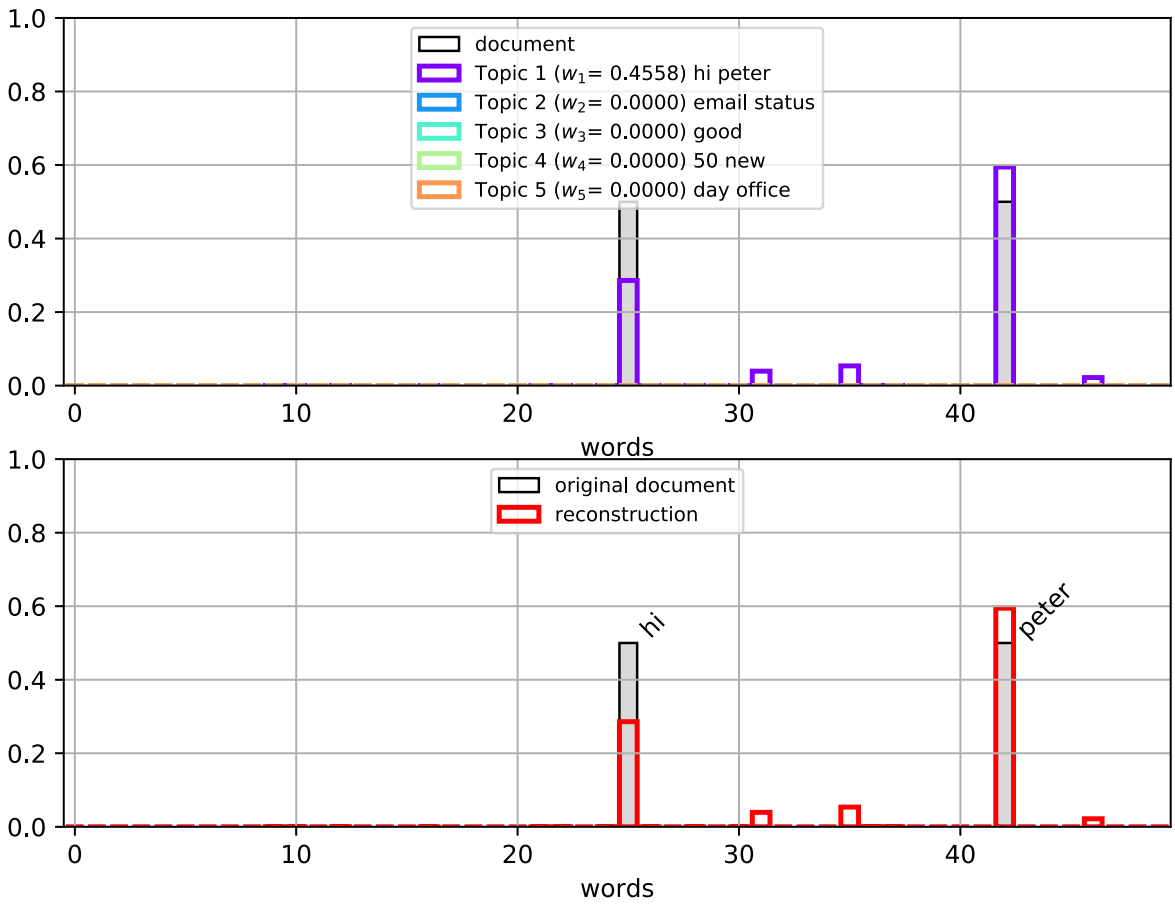


Document vector

- additive combination of topics

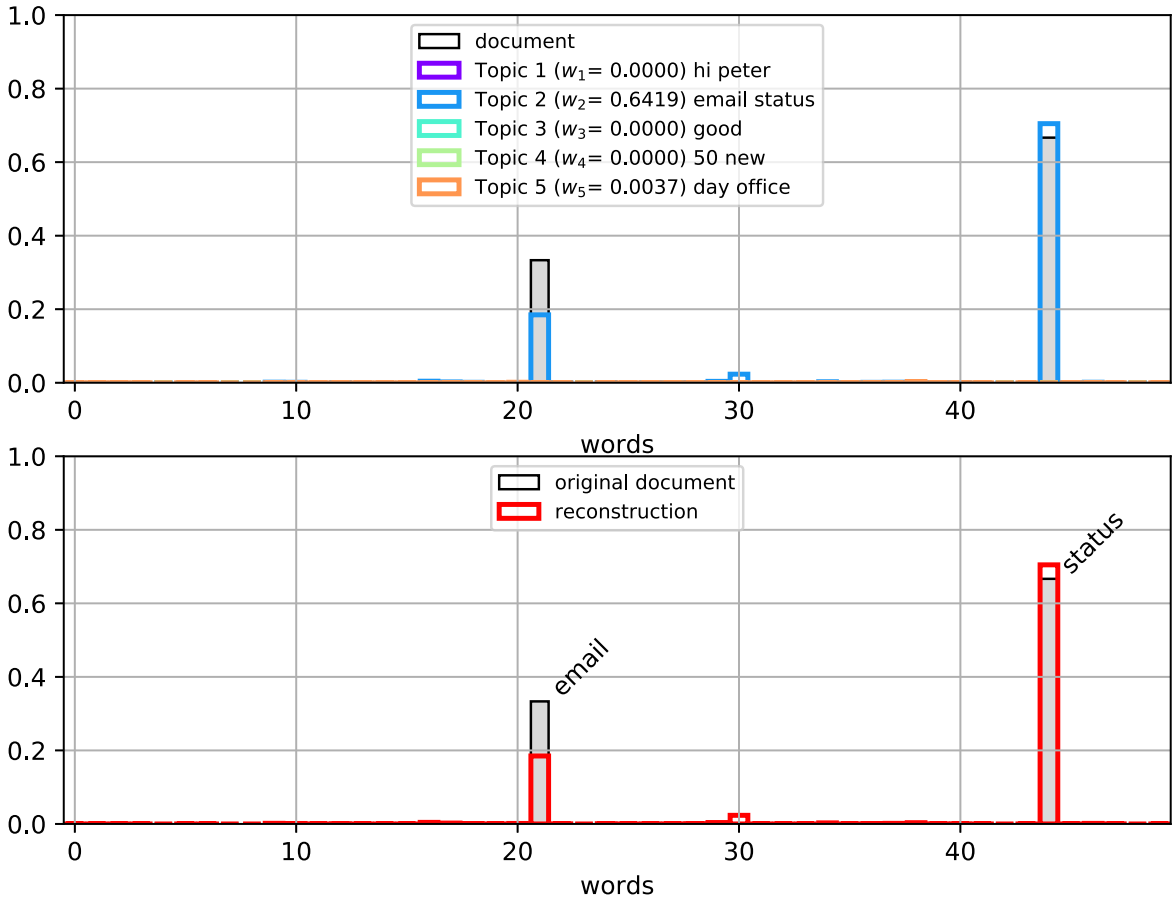
In [15]:
```
plot_doc_topic(Xtf[2,:], Wnmf[2,:], nmf, vocab)
```

Out[15]:



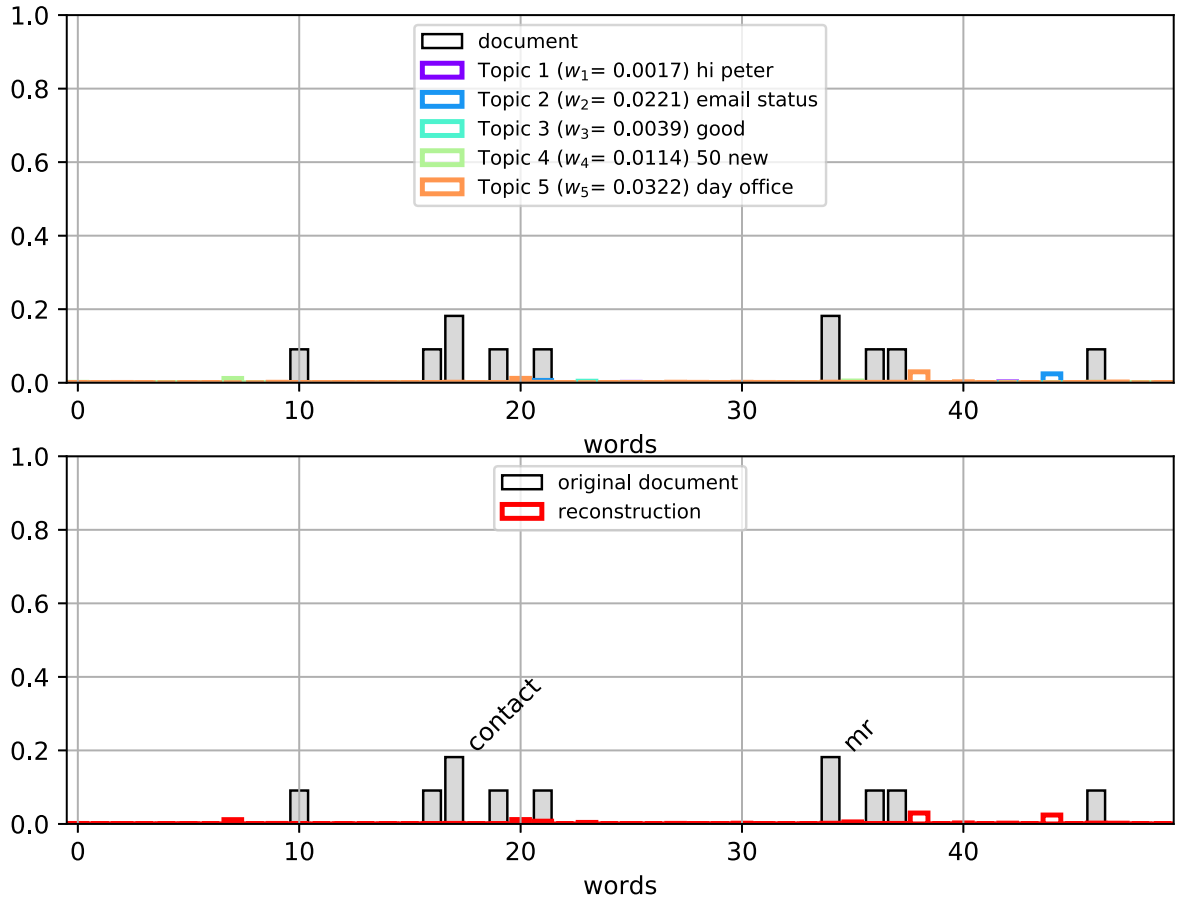
In [16]:
```
plot_doc_topic(Xtf[4,:], Wnmf[4,:], nmf, vocab)
```

Out[16]:



In [17]:
```
plot_doc_topic(Xtf[3,:], Wnmf[3,:], nmf, vocab)
```

Out[17]:

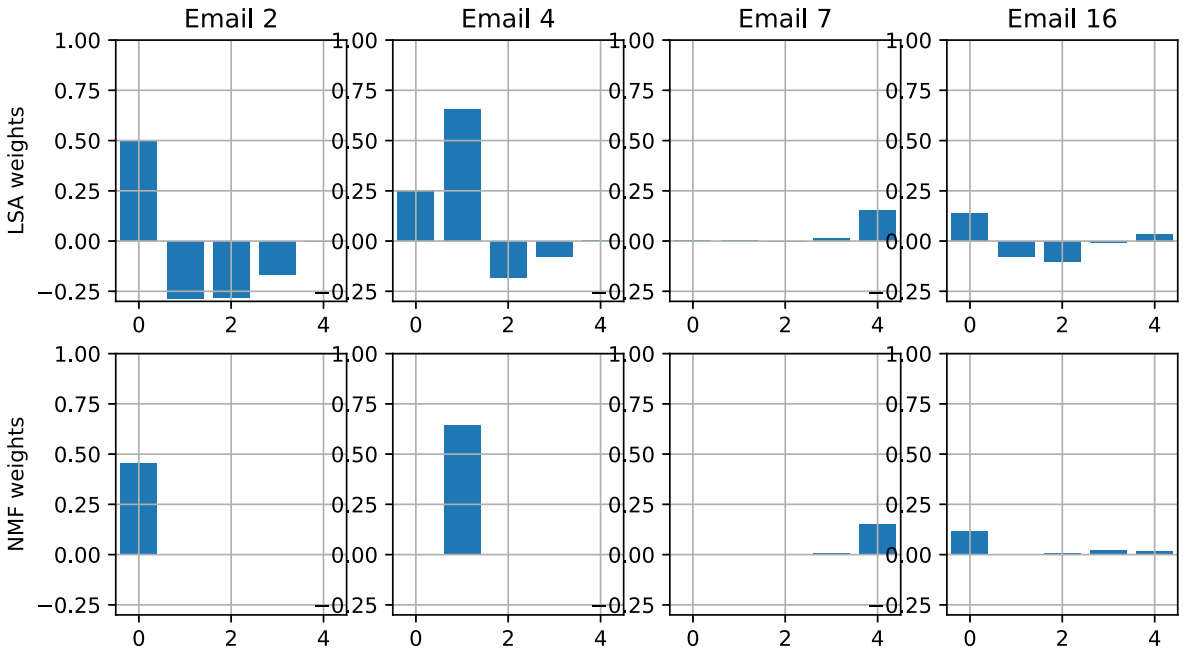


Sparseness

- For NMF representation, most topic weights for a document are zero.
  - this is called a *sparse* representation.
  - each document is only composed of a few topics.

In [19]:
```
spfig
```

Out[19]:

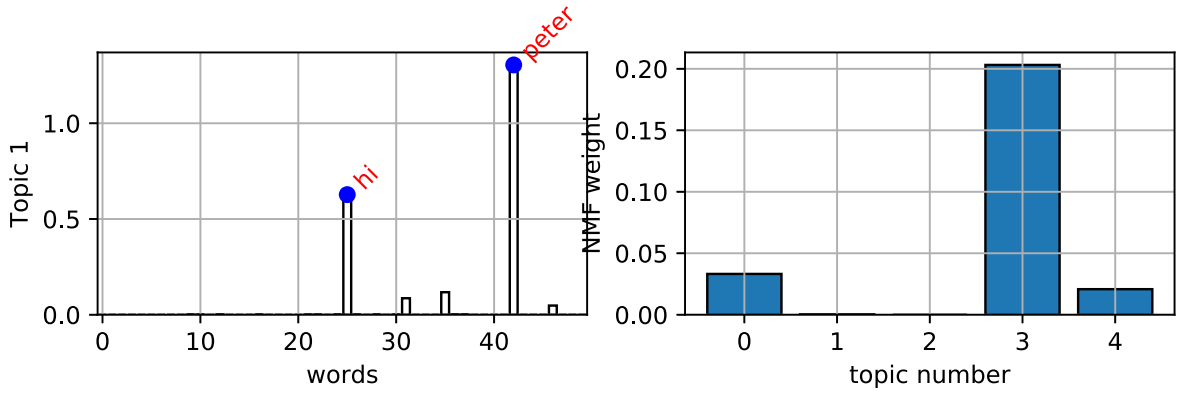


Problem with NMF

- *While the weights and component vectors are non-negative, NMF does not enforce them to be probabilities.*
- TF/TFIDF is a probabilistic model of words in a document
  - the vector of probabilities sums to 1
  - probabilities are between 0 and 1
- The NMF components and weights are difficult to interpret.

In [21]:
```
nfig
```

Out[21]:

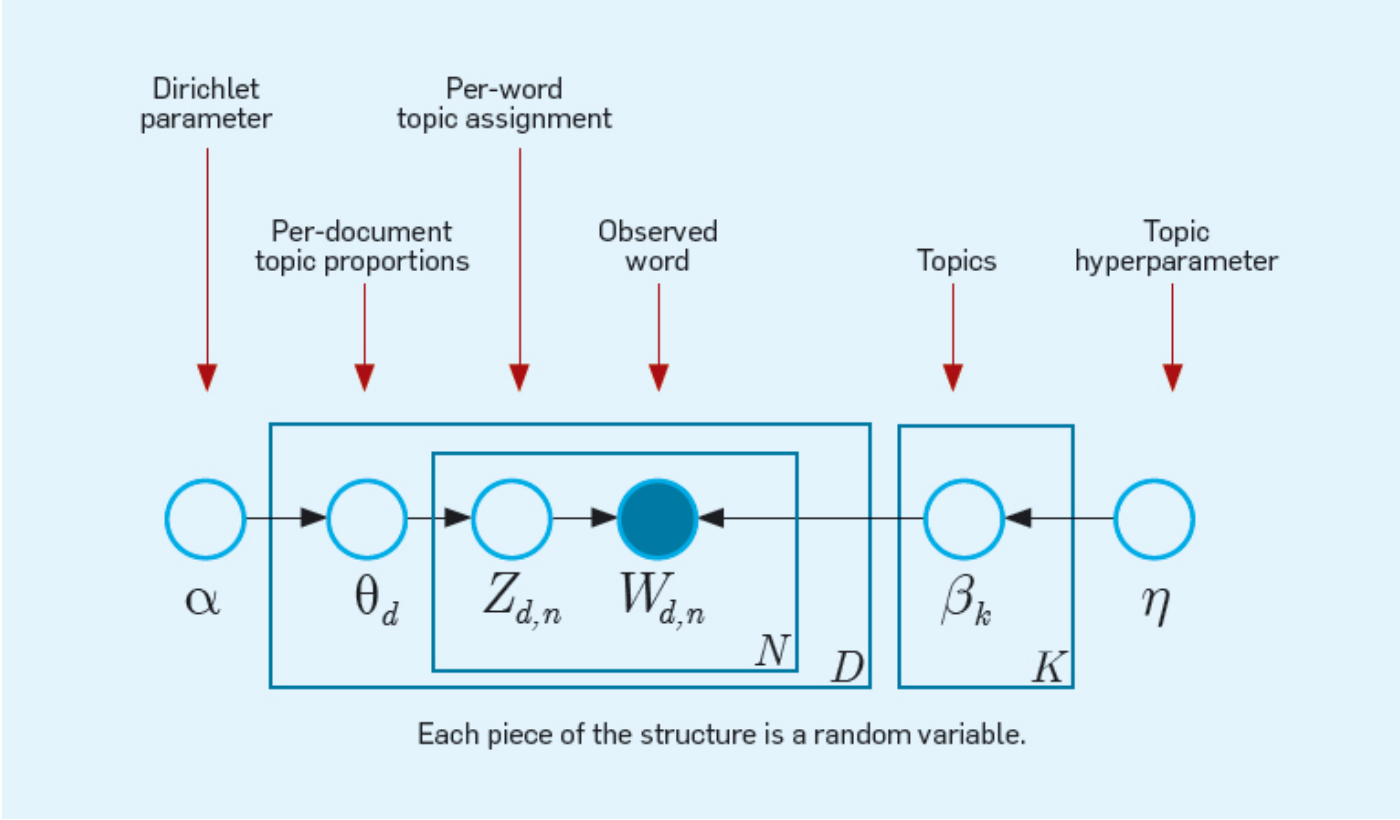


Latent Dirichlet Allocation (LDA)

- Use a generative probabilistic framework to model topics and documents.
- A document is composed of a mixture of topics.
  - Each topic has its own distribution of words (topic vector $\boldsymbol{\beta}_k$).
    - Topic vectors are shared among documents.
  - Each document has its own topic weighting.
    - The d-th document: $\hat{\mathbf{x}_d} = \sum_{k=1}^{K} \theta_{d,k} \boldsymbol{\beta}_k$
      - $\theta_{d,k}$ is the probability of the k-th topic occurring in the d-th document.
      - $\boldsymbol{\beta}_k$ is the topic vector for the k-th topic.

LDA graphical model

- Each node is a random variable.
- Plates (boxes) denote a vector of random variables.
  - The size is given in the bottom-right corner.



- LDA generative model
  1. For each topic $k = 1 \dots K$:
     A. Sample the topic vector $\boldsymbol{\beta}_k$ from a Dirichlet distribution.
  2. For each document $d = 1 \dots D$:
     A. Sample the topic weights θ_d from a Dirichlet distribution.
     B. For each word-position $n = 1 \dots N$:
        a. Sample a topic $z_{d,n}$.
        b. Sample a word $w_{d,n}$ from topic $z_{d,n}$.

LDA implementation

- Input X is the word counts from `CountVectorizer`.
- Important parameters:
  - `n_components` - The number of topics (K).
  - `doc_topic_prior` - Smoothing parameter (α) for the topic weights θ.
  - `topic_word_prior` - Smoothing parameter (η) for the topic vector β.
- Note: in the sklearn implementation,
  - the topic weights are not normalized on output.
  - the topic vectors are not normalized in `components_`
  - can be parallelized (`n_jobs`)

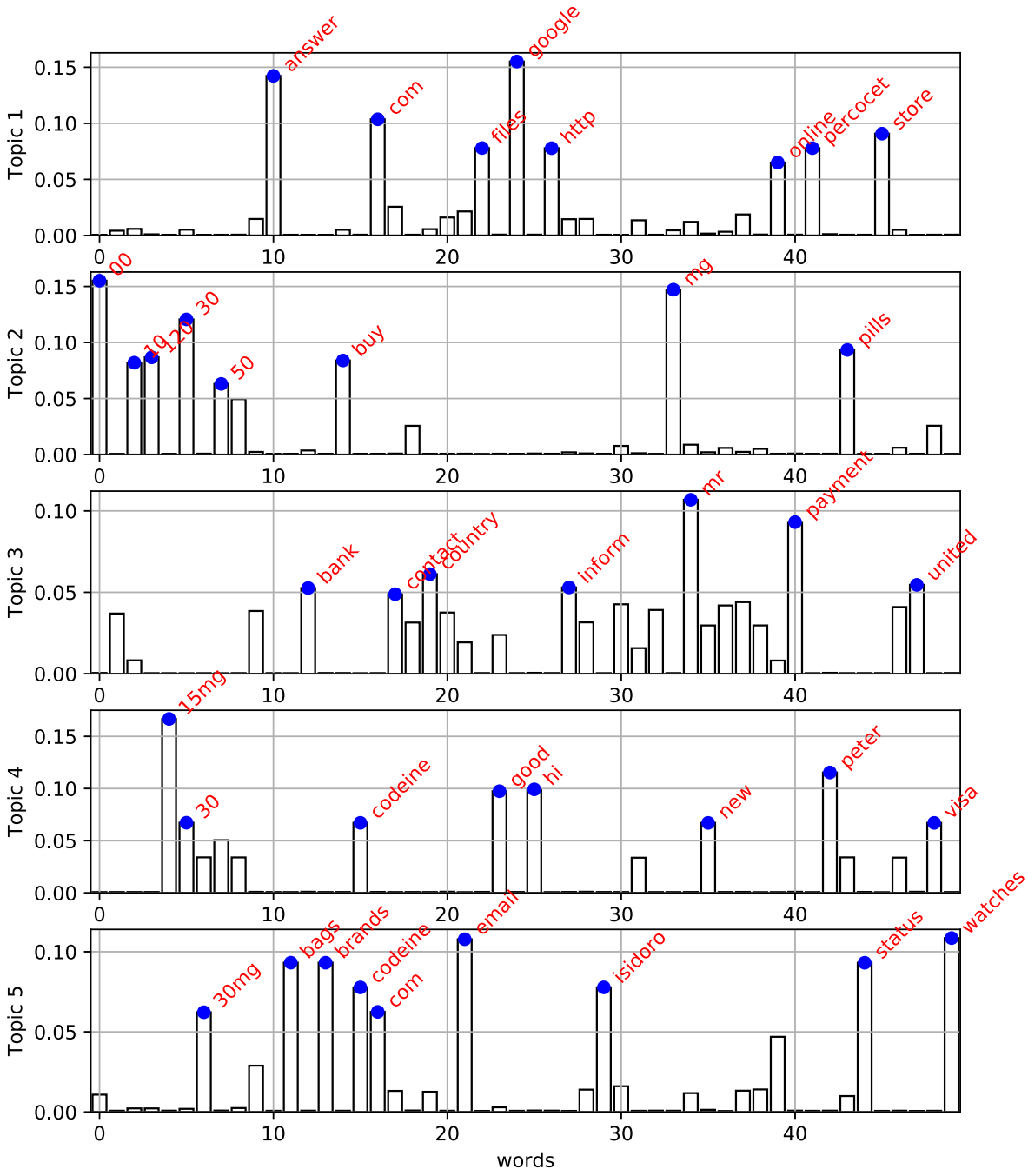
In [22]:
```
# fit LDA model
lda = decomposition.LatentDirichletAllocation(
    n_components=5,
    doc_topic_prior=0.01,
    topic_word_prior=0.01,
    random_state=4487, max_iter=25, n_jobs=-1)
# fit with X
Wlda = lda.fit_transform(X)

# normalize rows to be probabilities
Wlda /= sum(Wlda, axis=1)[:,newaxis]
```

```
/anaconda3/lib/python3.5/site-packages/sklearn/decomposition/online_lda.py:536
: DeprecationWarning: The default value for 'learning_method' will be changed
from 'online' to 'batch' in the release 0.20. This warning was introduced in 0
.18.
  DeprecationWarning)
```

- Topic vectors
  - Note that there is a small probability for each word (controlled by smoothing parameter η).

In [23]:
```
plot_topics(lda, vocab)
```

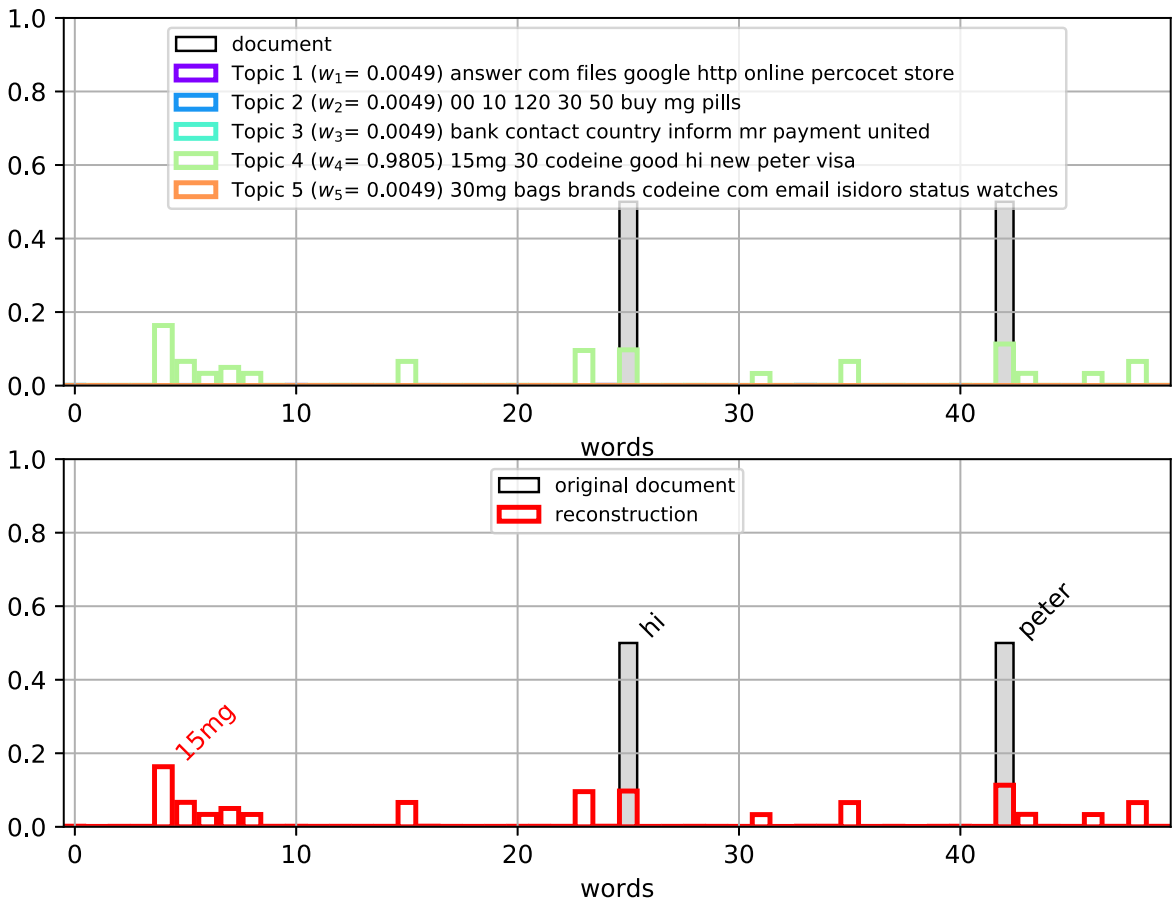
Out[23]:



- Document vector
  - Note the small probability for each topic (controlled by smoothing parameter α).

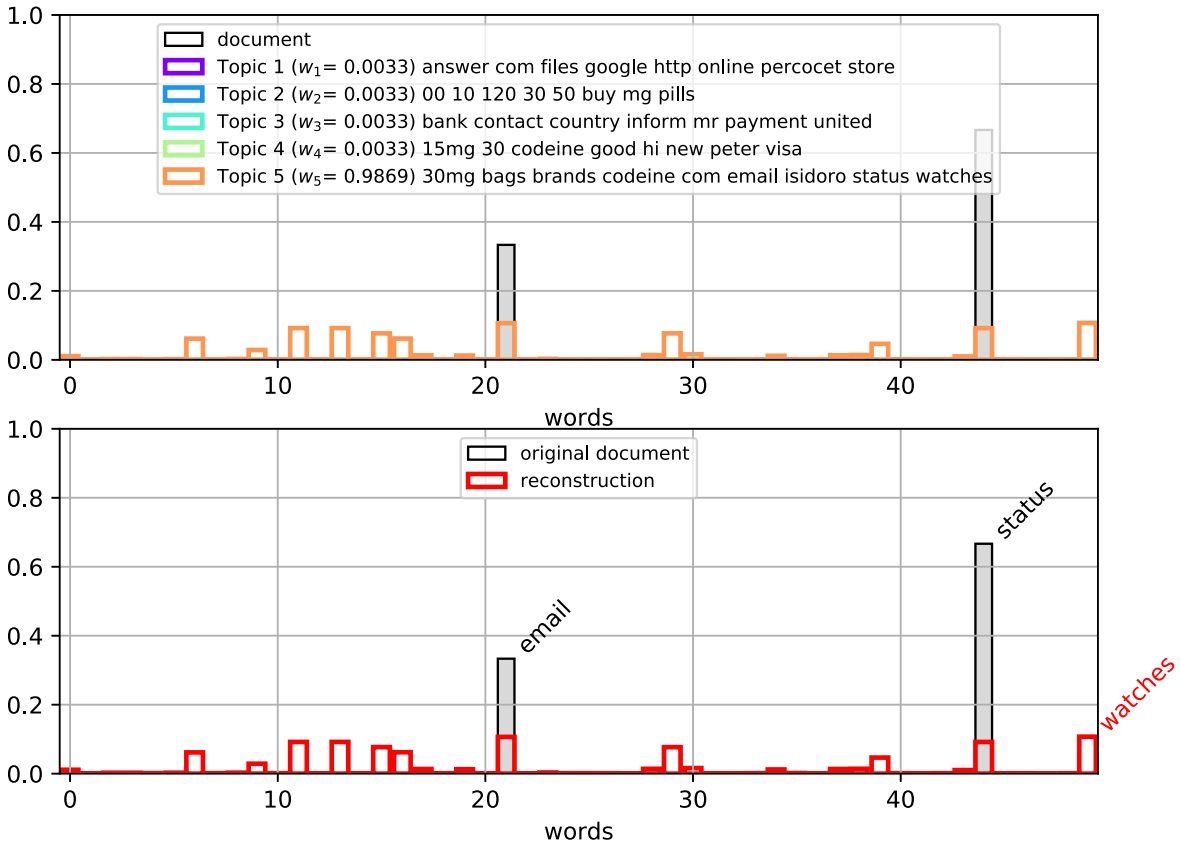
In [24]:
```
plot_doc_topic(Xtf[2,:], Wlda[2,:], lda, vocab)
```

Out[24]:



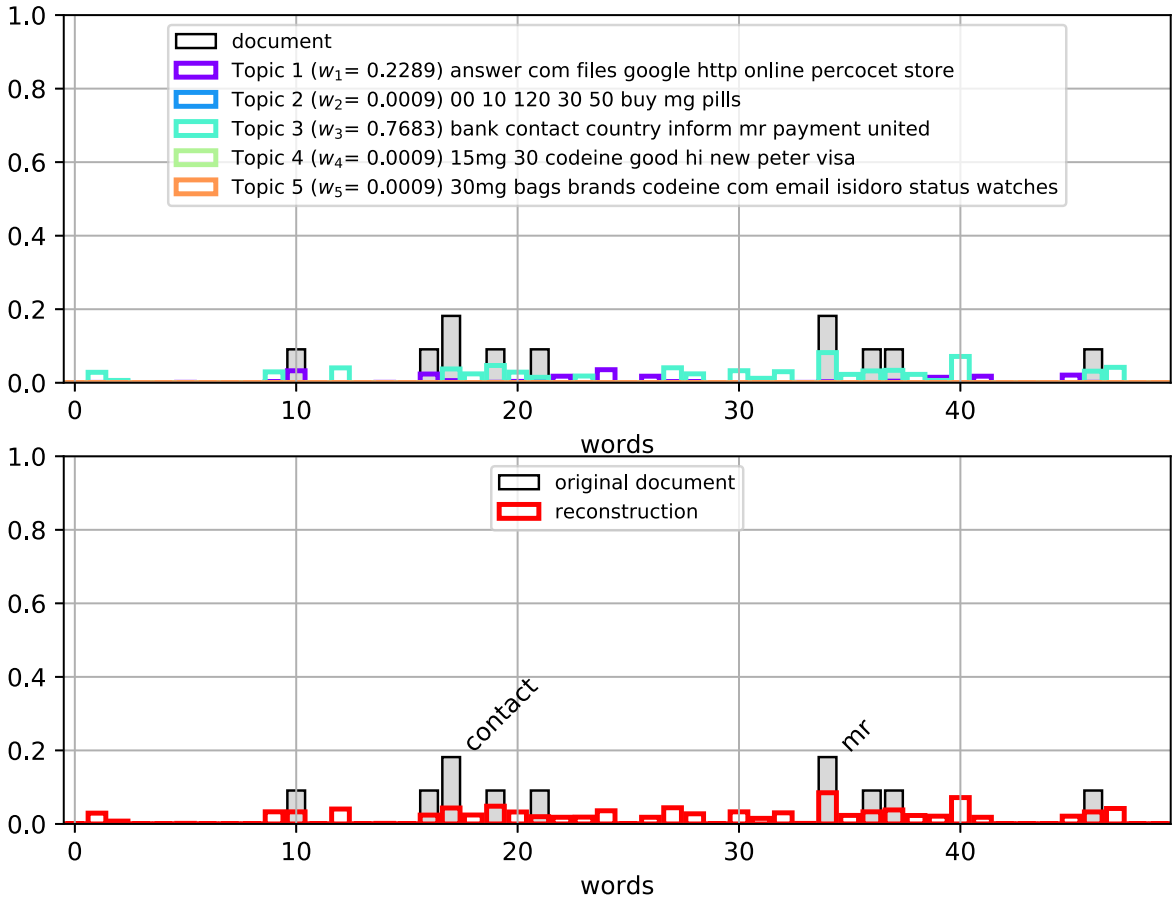
In [25]:
```
plot_doc_topic(Xtf[4,:], Wlda[4,:], lda, vocab)
```

Out[25]:



In [26]:
```
plot_doc_topic(Xtf[3,:], Wlda[3,:], lda, vocab)
```

Out[26]:

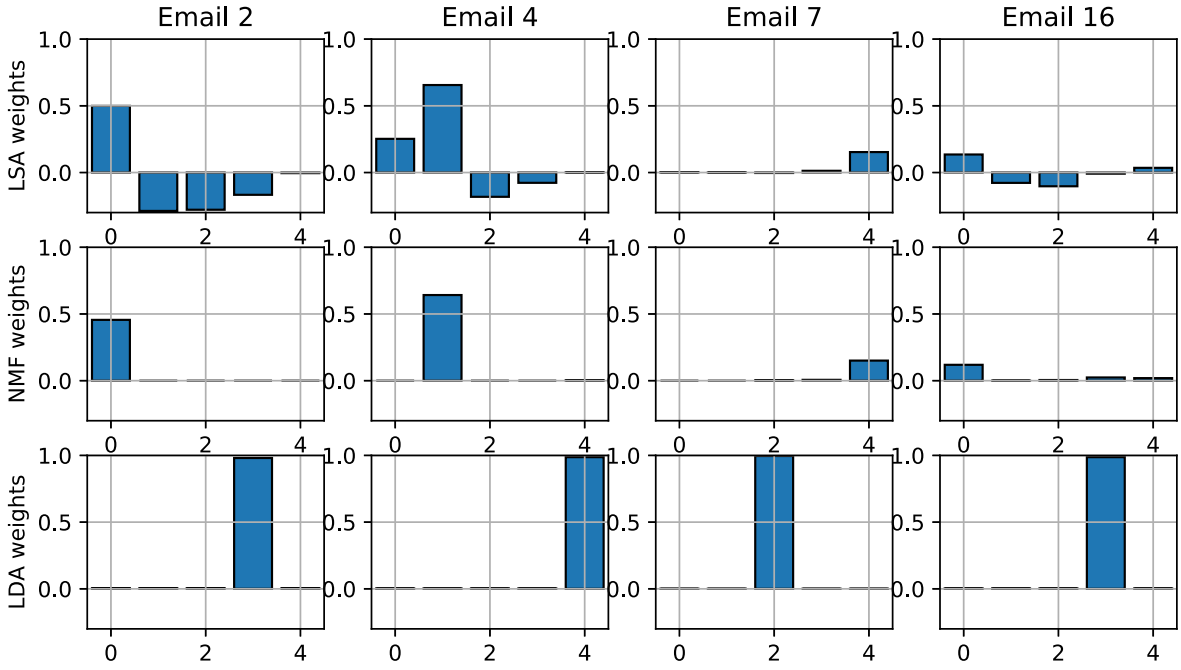


Comparison of topic weights

- LDA weights are probabilities.

In [28]:
```
compfig
```

Out[28]:



Linear Dimensionality Reduction - Summary

- **Goal:** given set of input vectors $\{\mathbf{x}_i\}_{i=1}^n$, with $\mathbf{x}_i \in \mathbb{R}^d$, represent each input vector as lower-dimensional vector $\mathbf{w}_i \in \mathbb{R}^p$.
  - Approximate $\mathbf{x}$ as a weighted sum of basis vectors $\mathbf{v}_j \in \mathbb{R}^d$
    - $\hat{x} = \sum_{j=1}^p w_j \mathbf{v}_j$
    - minimize the reconstruction error of $\hat{\mathbf{x}}$.
  - enables faster processing, or reduces noise.

Name	**Objective**	**Advantages**	**Disadvantages**
Principal component analysis (PCA)	minimize reconstruction error; preserve the most variance of data	- captures correlated dimensions, removes redundant dimensions, removes noise. - closed-form solution	- does not consider end goal (e.g., classification)
Random Projections	sample random basis vectors.	- fast. - preserves pairwise distances between points (up to accuracy factor).	- adds noise to the pairwise distances.
Fisher's Linear Discriminant (FLD)	maximize class separation	- preserves class separation	- requires class information
Latent Semantic Analysis (LSA)	minimize reconstruction error	- topic vectors have semantic meaning (co-occuring words) - closed-form solution	- topic weights and topic vectors can be negative - does not consider end goal (e.g., classification)
Non-negative Matrix Factorization (NMF)	minimize reconstruction error; non-negative weights and basis vectors.	- "additive" topic/parts model for text or images - sparse topic weights.	- solution requires iterative algorithm. - does not consider end goal (e.g., classification)
Latent Dirichlet Allocation (LDA)	document is a mixture of topics.	- generative probabilistic model. - robust when dataset size is small.	- inference/training can be slow for larger datasets.

Other things

- *Feature Normalization*
  - PCA and LDA are based on the covariance between input dimensions.
  - applying *per-feature* normalization will yield a different PCA result!
    - normalizing each input dimension changes the relative covariances.

In [31]:
```
nfig
```

Out[31]: