Topic Models and Latent Dirichlet Allocation

Daniel Lawson — University of Bristol

Lecture 08.2 (v2.0.0)

Signposting

- This block is about modelling Languages, containing:
  - Part 1: The ‘Bag of Words’ model,
  - Part 2: Latent Dirichlet Allocation.

Bag-of-words model

- The bag-of-words model is the simplest tool for Natural Language Processing. It takes a trivial form:
  - A **vocabulary** is created, consisting of the set of all words in all considered documents.
  - Each **document** is represented as a **feature vector** by counting the number of occurrences of each term (word).
  - Typically, documents are **sparse** as most words do not appear in most documents.

Notation

- **Terms** are indexed $t = 1 \ldots T$
- **Documents** are indexed $d = 1 \ldots D$
- **A document** X_d is a vector of term counts (sparsely stored)
- The **Corpus** $C = \{X_d\}_{d=1}^{D}$ is the set of all considered documents, and therefore contains all T terms

Python Bag-of-words

```
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
count = CountVectorizer()
docs = np.array([
'The sun is shining',
'The weather is sweet',
'The sun is shining and the weather is sweet'
])
bag = count.fit_transform(docs)
```

- See Python Machine Learning[1].

[1] Stevens, Kegelmeyer, Andrzejewsk and Buttler Exploring Topic Coherence over many models and many topics

Python Bag-of-words

```
>>> print(count.vocabulary_)
{'sweet': 4, 'shining': 2, 'weather': 6,
'and': 0, 'the': 5, 'is': 1, 'sun': 3}
>>> print(bag.toarray())
[[0 1 1 1 0 1 0]
 [0 1 0 0 1 1 1]
 [1 2 1 1 1 2 1]]
```

Word importance

- A popular measure of word relevancy is **term frequency-inverse document frequency** (**tf-idf**).
- tf-idf takes a very simple form:

$$\mathrm{tf}-\mathrm{idf}(t,d) = \mathrm{tf}(t,d) \times \mathrm{idf}(t,d)$$

- Where the term frequency $\mathrm{tf}(t,d) = X_d(t) / \sum_{t=1}^{T} X_d(t)$ is the frequency of term t in document d.
- The (log) inverse document frequency is:

$$\mathrm{idf} = \log\left(\frac{D}{1+n_d(t)}\right) = -\log\left(\frac{1+n_d(t)}{D}\right)$$

  - Where n is the total number of documents,
  - $n_d(t) = \sum_{d=1}^{n} \mathcal{I}(X_d(t) > 0)$ is the number of documents d that contain the term t.
  - The 1 is a smoothing term... (see Bayes in 7.1.2)

Interpreting tf-idf

- Clearly this is arbitrary, though based on a reasonable principle. . .
- TF accounts for the frequency within the document
- IDF assumes terms are *independent*, and ignores frequency:
  - The co-occurrence of two terms is the product of their probabilities, or the sum of their log probabilities
  - This ignores term frequency within each document
- This is therefore approximating $\Pr((t|d) \wedge (t \in d)) \log(\Pr(t \in d))$
- This can be rearranged into $\Pr(d|t) \propto \Pr(d, t)$,
- And resembles the elements of a **Mutual Information** measure:

$$(T, D) = \sum_t \sum_d p(t, d) \log \left(\frac{p(t, d)}{p(t)p(d)} \right).$$

Interpreting tf-idf

- The resemblance is meaningful, but not rigorous[2]
- Some hand-waving is required to get there:
  - $\text{tf} = \Pr(t|d) = X_d(t)/\sum_{t=1}^{T} X_d(t) \approx \frac{1+n_d(t)}{D}$ i.e. knowing the term tells you it is from one of the documents containing that term,
  - $\text{idf} = -\log(\Pr(d|t))$
  - $\Pr(d) = 1/D$
- The mutual information form can be reached by rearranging these sorts of statements
- It is not precise because different approximations are used in different elements
- And Mutual Information is a property of distributions, not of elements of that distribution.
- Very many other interpretations exist!
- These *hacks* can justified on robustness grounds.

[2]Stephen Robinson, Microsoft Research Understanding Inverse Document Frequency: On theoretical arguments for IDF

Python tf-idf

```
from sklearn.feature_extraction.text import TfidfTransformer
tfidf = TfidfTransformer(use_idf=True,
                         norm='l2',
                         smooth_idf=True)
np.set_printoptions(precision=2)
print(tfidf.fit_transform(count.fit_transform(docs)).toarray())
[[ 0.    0.43  0.56  0.56  0.    0.43  0.  ]
 [ 0.    0.43  0.    0.    0.56  0.43  0.56]
 [ 0.4   0.48  0.31  0.31  0.31  0.48  0.31]]
```

Alternative transforms

- tf-idf is arbitrary. It induces a useful feature space for comparisons. It ignores word **usefulness**.
- Alternatives include:
    - Cosine Similarity
    - Any other transformation, especially those with information-theory interpretations
    - feature extraction methods to understand classification importance
    - **Word2Vec**: Implemented in the package `gensim`.
    - **Doc2Vec**: Another option.
    - Modelling, e.g. **Latent Dirichlet Allocation**.

N-grams

- The previous analysis treats words as a “unit of inference”.
- It is instead possible to consider **N-grams**, that is, all **occurrences of (up-to) N characters**.
- Given enough data, it is possible to learn the words.
- This is valuable for modelling, e.g.:
  - Foreign languages: all unicode characters can be handled,
  - Non-languages such as computer code or byte strings, such as seen in binary executables,
  - Arbitrary factor sequences.
- They are typically stored efficiently (see **hashing** later in the course).
- The penalty is that:
  - larger corpora are required to obtain the same classification performance,
  - the feature space is dramatically larger,
  - word standardization cannot be used (see 7.2)

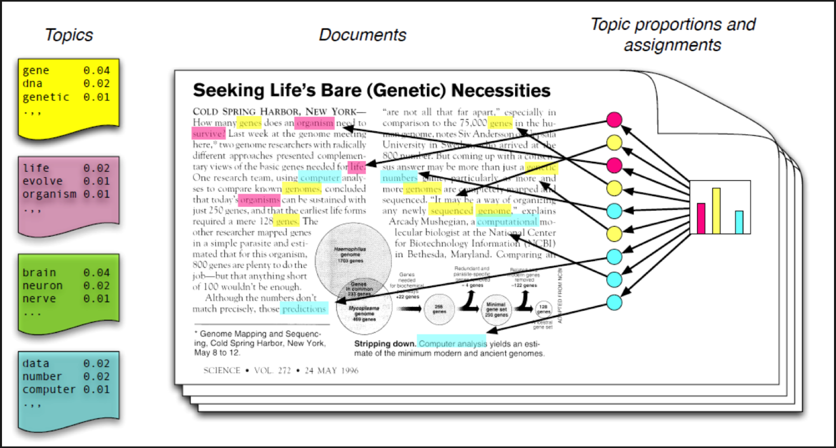
Beyond the bag of words

- The Bag-of-words is a **vector representation** of a set of documents.
  - i.e. a feature space embedding.
- But how can we use this? How do we **compare** documents?
  - We could perform **dimensionality reduction** via PCA,
  - **Distance metrics** such as Cosine Similarity,
  - etc.
- Or we can model the similarity. The most successful approach for this is **Latent Dirichlet Allocation** (LDA).

Modelling a Bag Of Words using Latent Dirichlet Allocation

- Each document is modelled as a **mixture** of topics,
- Each topic is modelled as a **distribution** over words,
- Some Bayesian modelling magic allows the documents to be a theoretically **infinite mixture** (see 04.1 - Nonparametrics).

LDA Motivation - The setup



gene	0.04
dna	0.02
genetic	0.01
,,,	

life	0.02
evolve	0.01
organism	0.01
,,,	

brain	0.04
neuron	0.02
nerve	0.01
...	

data	0.02
number	0.02
computer	0.01
,,,	

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

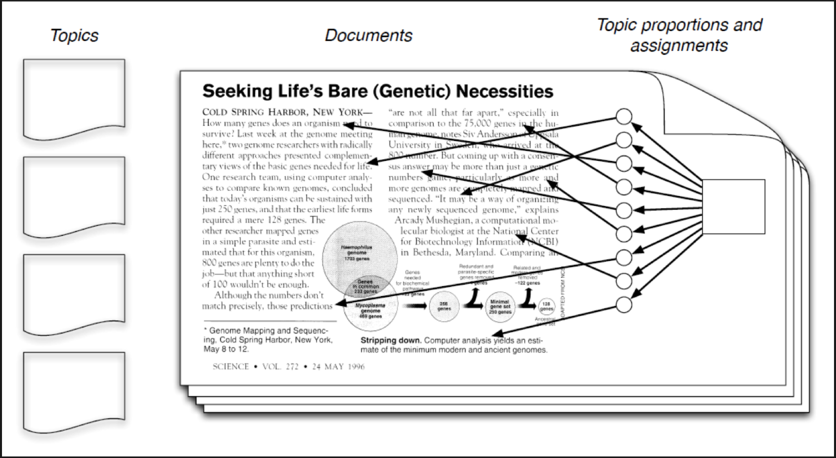
Haemophilus genome 1703 genes
Genes in common 233 genes
Mycoplasma genome 469 genes
Genes needed for biochemical pathways +22 genes
256 genes
Redundant and parasite-specific genes removed −4 genes
Minimal gene set 250 genes
Related and modern genes removed −122 genes
128 genes
Ancestral gene set
ADAPTED FROM NCBI

Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

SCIENCE • VOL. 272 • 24 MAY 1996

LDA Motivation - Data in Practice


Topics
Documents
Topic proportions and assignments
Haemophilus genome 1703 genes
Genes in common 233 genes
Mycoplasma genome 469 genes
Genes needed for biochemical pathways +22 genes
256 genes
Redundant and parasite-specific genes removed -4 genes
Minimal gene set 250 genes
Related and modern genes removed ~122 genes
128 genes
Ancestral gene set
ADAPTED FROM NCBI

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK— How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

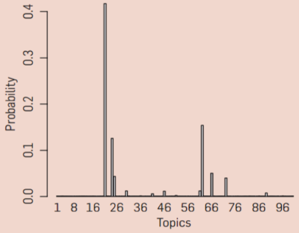
Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

SCIENCE • VOL. 272 • 24 MAY 1996

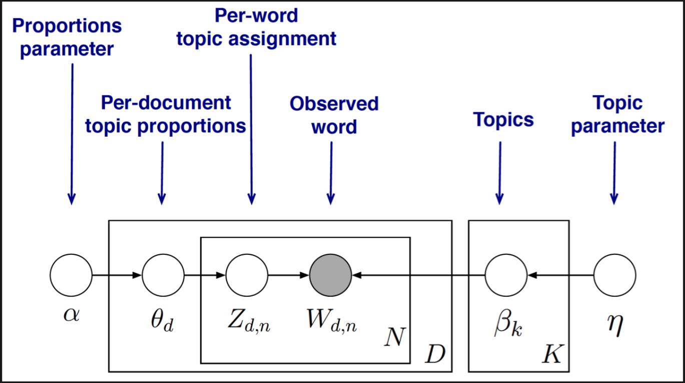
LDA Motivation - Example

The resulting output from an LDA model would be sets of topics containing keywords which would then be manually labeled. On the left are the inferred topic proportions for the example article from the pervious figure.



“Genetics”	**“Evolution”**	**“Disease”**	**“Computers”**
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

LDA Probabilistic Graphical Model



This is **plate notation** for **Bayesian Graphical Models**.

LDA Definition

- The overall **word distribution** is η, an N-vector.
- The overall **topic distribution** is α, a K-vector.
- Each **topic** k is described by a **word frequency** vector $\beta_k \sim \mathrm{Dirichlet}(\eta)$.
- Each **document** d is described by a **topic frequency** vector $\theta_d \sim \mathrm{Dirichlet}(\alpha)$.
- When generating word i from document d, we **generate a topic** $z_{di} \sim \mathrm{Multinomial}(\theta_d)$.
- And then **generate a word** $w_{di} \sim \mathrm{Multinomial}(\beta_d)$.

LDA properties

- Because it is a generative model, we can can ask it to simulate documents.
- These approaches are embarrassing:
  - in the sense that if you simulate from the model, it generates garbage,
  - because words are independent.
- They should be thought of instead as keyword generators.
- This is extremely useful for a variety of text categorisation tasks.
- It can operate:
  - supervised (where we insist that some documents have pre-defined topic distributions) or
  - unsupervised (where nothing is assumed apriori about topics).

LDA implementation

- LDA implementations[3] use a conjugate model (Multinomial distribution is conjugate to the Dirichlet prior).
- It uses Variational Bayes to write the problem as an optimisation problem.

[3]Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." Journal of machine Learning research 3.Jan (2003): 993-1022.

Further notes on LDA

- LDA models a matrix $Y = AX$, where:
  - Y is the data (N rows containing L word frequencies),
  - X are the topics (K rows containing L word frequencies) and
  - A is a mixture, (N rows containing K topics)
- This is a common problem called **matrix decomposition**.
- What makes LDA special is that **words are sparse**, meaning that there are many words but most words don't appear in most documents.
- You can run LDA on any problem of this type, but there are other approaches for dense data. (We return to **sparsity** later.)

Extensions

- We will not cover them, but if you work with document models you may want a more realistic model.
- Predictive text uses Markov Chains to predict $p(t(i)|d, t(i-1))$.
- Neural Networks generate arbitrary correlation structure, e.g.
  - Mathgen generates random papers,
  - Topic-RNN infers a topic model using a Neural Network.

Quantifying solutions

- There are many ways to quantify how good a particular LDA model is. The most popular are:
- **Perplexity**: the perplexity is $2^{-H(D)}$ where $H(D) = \sum_{t=1}^{T} \log(p(t|\theta_d))$
  - $p(t|\theta_d) = \sum_{v=1}^{V} \theta_d(v) p(t|v)$ uses the model-learned topics V for the (held out!) document d with topic distribution θ_d.
  - It is the entropy of term t (normally reported as the average per-word).
  - Perplexity is low (better) when each word appears in only one topic.
  - Perplexity is high when words are distributed across topics.
- **Coherence**: a measure of how often pairs of words appear together. there are two ways to examine this:
  - **intrinsic coherence**: called u_mass, this compares within a corpus.
  - **extrinsic coherence**: called c_v, this compares to some standard reference documents.
- Neither is particularly consistent with human judgement[4].

[4] Chang, Jonathan, Jordan Boyd-Graber, Sean Gerrish, Chong Wang and David M. Blei. 2009. Reading Tea Leaves: How Humans Interpret Topic Models.

Coherence

- The coherence is based on the score [5] (defined next):

$$Coherence(V) = \sum_{(t_i, t_j) \in V} score(t_i, t_j)$$

  - Where V is a topic, and t_i, t_j are word pairs.
- In both cases we use a **regulariser** ϵ.
  - $\epsilon = 1$ is natural but not obligatory.

[5] Stevens, Kegelmeyer, Andrzejewsk and Buttler Exploring Topic Coherence over many models and many topics

intrinsic coherence

- Using the score function:

$$u\_mass(v_i, v_j) = \log\left(\frac{p(v_i, v_j, \epsilon)}{p(v_i)p(v_j)}\right)$$

- i.e. we compare the probability that the words co-occur in a document with their relative frequencies.
- ϵ assigns non-zero weight to word pairs that do not occur together in a document.

extrinsic coherence

- Using the score function:

$$c\_v(v_i, v_j) = \log\left(\frac{D(v_i, v_j, \epsilon)}{D(v_j)}\right)$$

- where D counts documents that contain the word(s);
- i.e. we compare the frequency in which words co-occur in an external dataset, compared to their external frequency.

Reflection

- What is a bag of words, conceptually?
- What are the advantages of LDA over Bag of words?
- And vice-versa?
- Could you use SVD on a bag of words?
- Why would we use either, when empirical accuracy of neural-network approaches is higher?

References

- Bag-of-words: p259 Python Machine Learning (Raschka & Mirjalili, 2nd ed 2017)
- Topic Modeling and Latent Dirichlet Allocation: An Overview (Weifeng Li, Sagar Samtani and Hsinchun Chen)
- Stephen Robinson, Microsoft Research Understanding Inverse Document Frequency: On theoretical arguments for IDF
- Topic Modeling and Latent Dirichlet Allocation: An Overview (Weifeng Li, Sagar Samtani and Hsinchun Chen)
- Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation", Journal of machine Learning research 3.Jan (2003): 993-1022.