Outline

- Introduction to topic modelling
- Latent Semantic Analysis
- Latent Dirichlet Allocation
- Gensim
Topic modelling

- organize and understand large collections of documents
- text mining
- discover topical patterns in documents
- topic – group of words representing the information
- applications
  - recommender systems
  - document/book classification
  - bio-informatics (interpret biological data)
  - opinion/sentiment analysis
  - chatbots, topic tracking
  - text categorization
Recommender systems

- recommend the best product for user
- clusters of users, based on preference
- clusters of products
- Netflix prize
Bio-informatics

- categorize patients to risk groups, based on text protocols
- detect common genomic features, based on gene sequence data
- group drugs by diagnosis

![Diagram showing topic identification and modeling]
Latent Semantic Analysis

- vector representation of documents
- compare by vector distance
- document = bag of words
- topic = set of words
- applications:
  - data clustering, document classification
  - term relations (synonymy, polysemy)
  - cross language document retrieval
  - word relations in text
  - similarity in multi choice questions
  - prior art in patents
LSA – step 1

- count **term-document matrix** (word frequency in documents)
- rows = words, columns = documents
- **sparse matrix**

<table>
<thead>
<tr>
<th>Terms</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
<th>M6</th>
<th>M7</th>
<th>M8</th>
<th>M9</th>
<th>M10</th>
<th>M11</th>
<th>M12</th>
<th>M13</th>
<th>M14</th>
</tr>
</thead>
<tbody>
<tr>
<td>abnormalities</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>age</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>behavior</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>blood</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>close</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>culture</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>depressed</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>discharge</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>disease</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>fast</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>generation</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>oestrogen</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pressure</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>rats</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>respect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rise</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>study</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
LSA – step 2

- weighting matrix elements
- most popular tf–idf
- term occurring in many documents is not interesting for analysis
LSA – step 3

- **Singular Value Decomposition**
- matrix factorization (reduce dimensions, throw away noise)
- cluster close vectors (documents and terms)
Latent Dirichlet Allocation

- statistical model
- each document is a mix of topics
- LDA discovers topics and their ratio
- each word in document was generated by one of the topics
- applications:
  - topic relations
  - content recommendation
  - group/community overlapping
  - document topic changes
  - genetics (ancestral populations)
Example

Document 1: I like to eat broccoli and bananas.
Document 2: I ate a banana and spinach smoothie for breakfast.
Document 3: Chinchillas and kittens are cute.
Document 4: My sister adopted a kitten yesterday.
Document 5: Look at this cute hamster munching on a piece of broccoli.

Example

Topic A: 30% broccoli, 15% bananas, 10% breakfast, 10% munching
Topic B: 20% chinchillas, 20% kittens, 20% cute, 15% hamster

Example

Document 1 and 2: 100% Topic A
Document 3 and 4: 100% Topic B
Document 5: 60% Topic A, 40% Topic B
LDA process

- pick fixed number of topics
- for each document, randomly assign topic to each word
- improve, for each document \( d \):
  - for each word \( w \) and topic \( t \) count:
    - all topic assignments are correct, except for current word
    - \( p(topic \ t | document \ d) \) – how many words in document have topic?
    - \( p(word \ w | topic \ t) \) – how many assignments to topic for word?
    - new topic: probability \( p(topic \ t | document \ d) \times p(word \ w | topic \ t) \)
- repeat and reach almost steady state
Topic Labeling

represent topic with human-friendly label
- top N words from the list
- find Wikipedia article based on word list
- document summarization from topic documents
```python
>>> from gensim import corpora, models, similarities

>>> # Load corpus iterator from a Matrix Market file on disk.
>>> corpus = corpora.MmCorpus('/path/to/corpus.mm')

>>> # Initialize Latent Semantic Indexing with 200 dimensions.
>>> lsi = models.LsiModel(corpus, num_topics=200)

>>> # Convert another corpus to the latent space and index it.
>>> index = similarities.MatrixSimilarity(lsi[another_corpus])

>>> # Compute similarity of a query vs. indexed documents
>>> sims = index[query]
```
Gensim – LSA

gensim.models.lsimodel.LsiModel(corpus=None, num_topics=200, id2word=None, chunksize=20000, decay=1.0, distributed=False, onepass=True, power_iters=2, extra_samples=100)

- chunksize – number of documents in memory (more documents, more memory)
- decay – newly added documents are more important?
- power_iters – more iterations improve accuracy, but lower performance
- onepass – False to use multi-pass algorithm, for static data increase accuracy
Gensim – LDA

gensim.models.ldamodel.LdaModel(corpus=None, num_topics=100, id2word=None, distributed=False, chunksize=2000, passes=1, update_every=1, alpha='symmetric', eta=None, decay=0.5, offset=1.0, eval_every=10, iterations=50, gamma_threshold=0.001, minimum_probability=0.01, random_state=None, ns_conf=None, minimum_phi_value=0.01, per_word_topics=False)

- chunksize – number of documents in memory (more documents, more memory)
- update_every – number of chunks before moving to next step
  - chunksize=100k, update_every=1 equals to chunksize=50k, update_every=2 (saves memory)
- decay – newly added documents are more important?
- alpha, eta – preset expected topics and word probability for start
- eval_every – log perplexity is estimated after x updates (lower number, slower training)
References I


