CSCI E-82
Advanced Machine Learning, Data Mining & Artificial Intelligence
Lecture 3

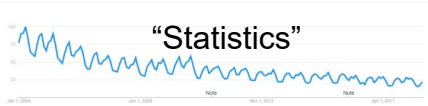
Text Mining (cont.)
Frequent Pattern Mining

Peter V. Henstock
Fall 2018

HW2

Homework 2

- Level 1: Word = concept
  - Find individual words and plot over time
    - Google trends



    - How do you define a word? Morphology?
  - Issues of normalization for
    - Document length
    - Number of documents per year

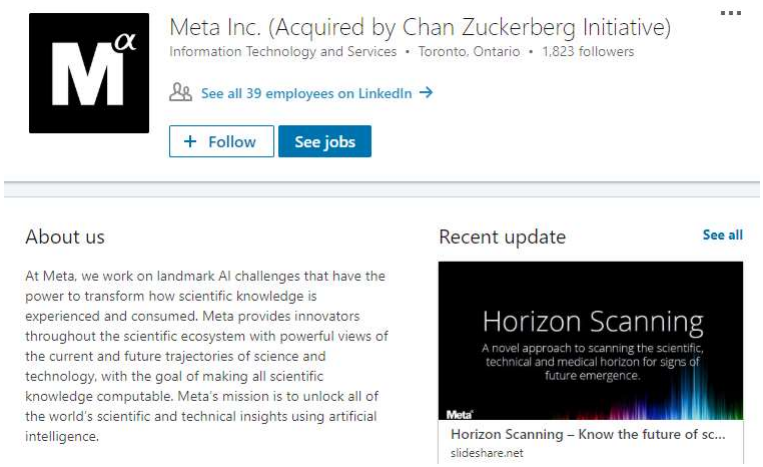
Homework 2

- Level 2a: Compound word option
  - “neural network” != “neural” or “network”
  - How can you find multi-word entities?
    - Frequent pattern mining idea
    - Probabilistic approaches
    - Part of speech information
    - TopMine, etc.

Homework 2

- Level 2b: Concept option
  - Prob. = probability = P = chance ~ likelihood
  - Synonym lists idea, or
  - Create a numeric representation of words
    - Word2vec or Latent Dirichlet Allocation or doc2vec
  - Use numeric representation + distances to define concepts through clustering, etc.
  - PCA for words → Latent Semantic Indexing
  - Clustering
  - Latent Dirichlet Allocation

Meta Inc.



- https://www.linkedin.com/company/sciencescape/
- https://beta.meta.org/account/reserve/

Text Mining

Natural Language Processing

- Hybrid field:
  - Computer science
  - Artificial Intelligence
  - Linguistics
- Goals:
  - Perform tasks using ‘natural’ language
  - Provide question/answering support
  - Characterize sentiment
  - Find and extract information

Are you using NLP?

- Spell checking
- Thesaurus
- Google searches
- Parsing web pages
- Classifying documents
- Sentiment analysis
- Yelp review verification
- Siri or Alexa
- Chatbots

Why NLP is key

- Can we 'think' without language?
  - Inner voice
- Not just a massive data source
  - Deliberate transmission of information
  - Symbolic signaling with words ~ concepts
  - 20K words typically is large vocabulary
  - Generally a continuous stream

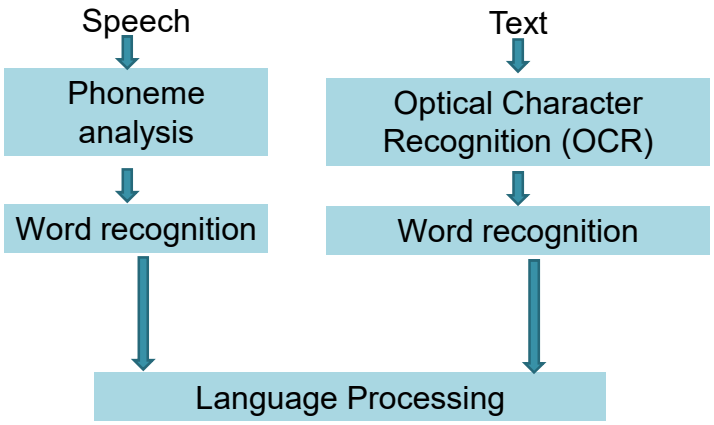
Why is NLP difficult?

Why is NLP difficult? Ambiguity

- Multiple word meanings

- Reference or Anaphora
  - The music stopped and that upset everyone.
  - Before it could fly, it was a worm
  - This one is better than that one.
- Need context or common sense to make the correct interpretation

Language Inputs



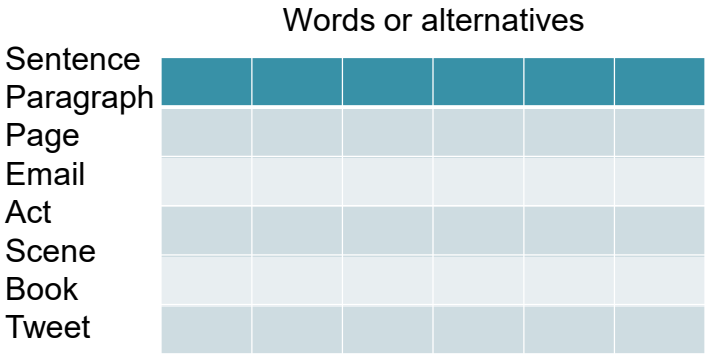
Linguistics Pyramid

- Pragmatics
- Sociolinguistics
- Semantics
- Syntax
- Morphology
- Lexical
- Phonology

A Text Analysis Strategy

- Convert unstructured text to numbers
- Produce a large table:
  - Row~document Column~word
- Apply standard approaches thereafter
  - Dimensionality Reduction
  - Classification
  - Clustering

What are the rows and columns?



Approaches to Text Mining

- Bag of words
  - Randomly ordered collection of words
  - {hello, my, name, is, peter}
- N-gram
  - Order within each set of N adjacent units but random N-grams
  - {hello my, my name, name is, is peter}
- Shallow parsing
  - Recognize subset of subject-object-verb or other relationships
    - Subject=Name
    - Verb=is
    - Predicate Nominative=Peter

Workflow for Bag of Words

- Extract the sentence/paragraph/page
- Remove the punctuation
  - Remove capitalization? (maybe)
- Extract the words
- Apply [Brill] Part of Speech tagger?
- Remove morphological endings
- Remove stop words

Brill Tagger

- Developed Eric Brill in 1992
- Supervised learning approach to label each word with a part of speech
- Uses a dictionary
- Uses the word structure to guess if it's not in the dictionary
- Set of rules

Brill Tagger

- Set of rules for POS
  - IN = preposition
  - NN = noun
  - DT = determinant
- Example Rule:
  - "IN NN WDPREVTAG DT while"
  - *While* defaults to a preposition (IN)
  - Change from preposition to noun (NN) if:
    - WDPREVTAG previous word is DT (a,the)
  - *Once in a while* has *while* as NN

Stemmers or Lemmatization

- Morphology = modification of a word
- Swim → swims, swimming, swam, etc.
- Peach→ peachy, peaches
- Amylase → β-amylase

- Goal: swimming, swims → swim
- Porter Stemmer is one of most common
  - Arguably pretty bad

- Some stemmers require part of speech
  - Noun plurals are good for nouns only

All words are not created equal

- Clustering documents are not likely to be influenced by filler words like “a”, “the”, “in”, etc.
- Useless words are usually removed
- Called “stop words”

All words are not created equal

- Google search
  - Toto, I've a feeling we're not in Kansas anymore
- Which words are most important?
- Google search: Toto → rock group
- Google search: Toto + feeling → toilet
- Google searches:
  - Kansas+ feeling → quote
  - Toto + Kansas → quote

All words are not created equal

- Google search
  - Toto, I've a feeling we're not in Kansas anymore
- Which words are most important?
- Basis of search or "information retrieval" across all languages
- How can we create such a rule?

TFIDF

- Term Frequency Inverse Document Frequency
- TF = fraction of document for that term
  - If corpus contains “Toto” 3 times in 400 documents, then 3/400
- IDF= ln(#doc / [#docs containing “Toto”])
  - 100,000 docs but 50 with Toto → ln(2000)
- TFIDF = TF * IDF

Converting text to tables

- Frequency x: very useful
- Log frequency: log(x+1)
- Binary occurrence: 0 or 1
  - Word appears in context or not
- TFIDF:
  - Term Frequency Inverse Document Frequency
  - Normalized by #times occurs in all contexts
  - Very useful for querying
  - x/log[(#contexts+1)/ contexts_containing_term]
- BM25: (k+1)*x/ [x+ k] for param k
  - Parameterize between frequency & TFIDF

Information Retrieval

- Take the search terms
- Sum the TFIDF over each search term
- Use score to rank order the documents
- But there are many variations
- PageRank (Google) uses links as well

Clustering Documents

- Method options:
  - Use the bag of words approach
    - Capitalization? Stemmer, Etc.
  - Use N-gram approach
    - Capitalization? Stemmer, Etc.
- Feature creation
  - Frequency counts of words per document
  - Binary presence of words in document
  - TFIDF option (not usually used for N-gram)

Example for sentences

- To be or not to be. That is the question
- {To be or not to be.} {That is the question}
  - Create sentences
- {to be or not to be} {that is the question}
  - Kill the punctuation and capitalization
- {to be or not to be} {that be the question}
  - Stemmer to get rid of tense information
- {be not be} {be question}
  - Remove stop words {or, to, the, that}

Entry	be	not	question
sentence1	2	1	0
sentence2	1	0	1

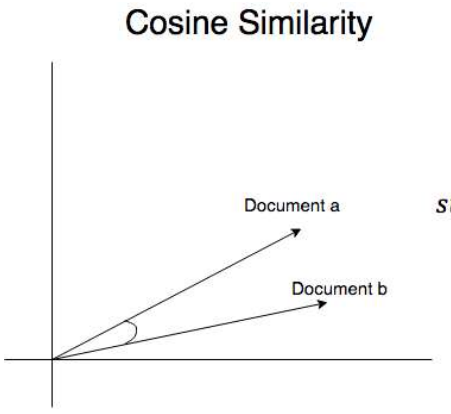
Words → features for paradigmatic

- Examine "contexts"
  - Neighboring words
  - Sentence
  - Paragraph
  - Section
  - Document
- Create vector for each word 1…M
  - Counts of occurrences of each word in a given context across corpus
  - $[x_1, x_2, \ldots x_M]$ per context
  - where x_i = #times "horse" occurs in context
- Closest words: paradigmatic relationship

Distance Metrics

- We have reduced language to vector representations
- We can compared vectors using distance metrics on raw or transformed vectors
- Cosine similarity: $\sum_{i=1}^{n} A_i B_i / [\sqrt{\sum A_i^2}\sqrt{\sum B_i^2}]$
- Euclidian distance
- Can now cluster the results

Cosine Similarity



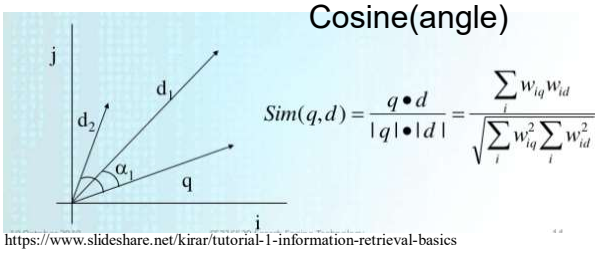
$$sim(a,b) = cos\theta = \frac{\vec{a}.\vec{b}}{\| \vec{a} \| \| \vec{b} \|}$$

Vector-Space Model

	Once	upon	a	midnight	journey	while	I	ponder
Doc1	1	1	9	2	3	2	3	2
Doc2	0	0	4	0	1	3	5	0
Doc3	0	0	2	0	1	0	0	0
…								
DocN	0	0	9	0	1	0	0	0

	Once	upon	a	midnight	journey	while	I	ponder
query	0	0	2	2	1	0	3	0

Cosine(angle)

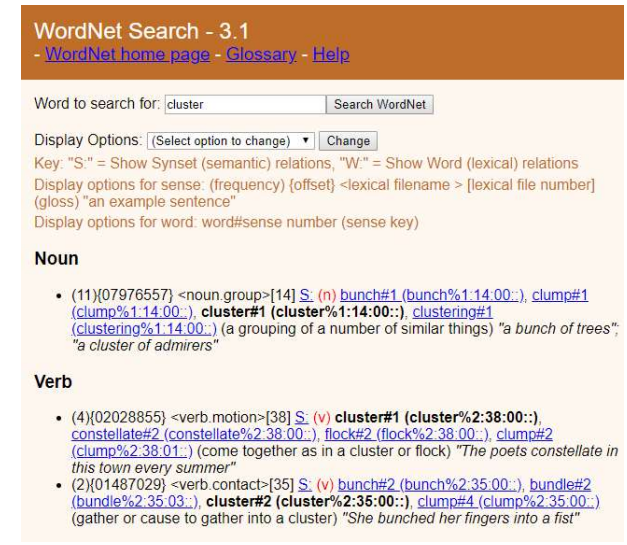


https://www.slideshare.net/kirar/tutorial-1-information-retrieval-basics

LSA & LSI

Synonym Lists

- https://wordnet.princeton.edu/



WordNet Search - 3.1
- WordNet home page - Glossary - Help

Word to search for: cluster Search WordNet

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (frequency) {offset} <lexical filename > [lexical file number] (gloss) "an example sentence"

Display options for word: word#sense number (sense key)

Noun

- (11){07976557} <noun.group>[14] S: (n) bunch#1 (bunch%1:14:00::), clump#1 (clump%1:14:00::), **cluster#1 (cluster%1:14:00::)**, clustering#1 (clustering%1:14:00::) (a grouping of a number of similar things) *"a bunch of trees"; "a cluster of admirers"*

Verb

- (4){02028855} <verb.motion>[38] S: (v) **cluster#1 (cluster%2:38:00::)**, constellate#2 (constellate%2:38:00::), flock#2 (flock%2:38:00::), clump#2 (clump%2:38:01::) (come together as in a cluster or flock) *"The poets constellate in this town every summer"*
- (2){01487029} <verb.contact>[35] S: (v) bunch#2 (bunch%2:35:00::), bundle#2 (bundle%2:35:03::), **cluster#2 (cluster%2:35:00::)**, clump#4 (clump%2:35:00::) (gather or cause to gather into a cluster) *"She bunched her fingers into a fist"*

WordNet Search - 3.1
- WordNet home page - Glossary - Help

Word to search for: software Search WordNet

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **software**, software program, computer software, software system, software package, package ((computer science) written programs or procedures or rules and associated documentation pertaining to the operation of a computer system and that are stored in read/write memory) *"the market for software is expected to expand"*
  - *direct hyponym* / *full hyponym*
    - S: (n) plug-in, add-on (a software module that extends or enhances the capabilities of an existing application)
    - S: (n) alpha software (a first release of a software product that is usually tested only by the developers)
    - S: (n) authoring language (software that can be used to develop interactive computer programs without the technically demanding task of computer programming)

<3 more pages of this>

  - *domain category*
    - S: (n) computer science, computing (the branch of engineering science that studies (with the aid of computers) computable processes and structures)
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - S: (n) code, computer code ((computer science) the symbolic arrangement of data or instructions in a computer program or the set of such instructions)
  - ***antonym***
    - W: (n) hardware [Opposed to: software] ((computer science) the mechanical, magnetic, electronic, and electrical components making up a computer system)

Word Meanings

- Word ~ meaning or conceptual idea
  - Difficult for computers to utilize
- Synonyms and ontologies
  - WordNet (http://wordnet.princeton.edu)
  - Still incomplete such as slang or technical
  - Nuance of usages are difficult to capture

Latent Semantic Indexing

- Compute SVD on matrix $A = S\Sigma U^T$
- Choose top k eigenvalues and eigenvectors
  - Choose 2 for example

$$S_2 = \begin{bmatrix} romeo & -0.396 & 0.280 \\ juliet & -0.314 & 0.450 \\ happy & -0.178 & 0.269 \\ dagger & -0.438 & 0.369 \\ live & -0.264 & -0.346 \\ die & -0.524 & -0.246 \\ free & -0.264 & -0.346 \\ new\text{-}hampshire & -0.326 & -0.460 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 2.285 & 0 \\ 0 & 2.010 \end{bmatrix} \quad U_2^T = \begin{bmatrix} -0.311 & -0.407 & -0.594 & -0.603 & -0.143 \\ 0.363 & 0.541 & 0.200 & -0.695 & -0.229 \end{bmatrix}$$

- Map terms using $S_2\Sigma_2$
- Map documents using ΣU_2^T
- Romeo = [-0.905 0.563] ^{T}
- D1 = [-0.711 0.730]^T

Latent Semantic Indexing

- http://webhome.cs.uvic.ca/~thomo/svd.pdf
- Matrix A =

	d_1	d_2	d_3	d_4	d_5
romeo	1	0	1	0	0
juliet	1	1	0	0	0
happy	0	1	0	0	0
dagger	0	1	1	0	0
live	0	0	0	1	0
die	0	0	1	1	0
free	0	0	0	1	0
new-hampshire	0	0	0	1	1

- Compute SVD on matrix A = $S\Sigma U^T$
- Choose top k eigenvalues and eigenvectors
  - Choose 2 for example

$$S_2 = \begin{array}{r} \textit{romeo} \\ \textit{juliet} \\ \textit{happy} \\ \textit{dagger} \\ \textit{live} \\ \textit{die} \\ \textit{free} \\ \textit{new-hampshire} \end{array} \begin{bmatrix} -0.396 & 0.280 \\ -0.314 & 0.450 \\ -0.178 & 0.269 \\ -0.438 & 0.369 \\ -0.264 & -0.346 \\ -0.524 & -0.246 \\ -0.264 & -0.346 \\ -0.326 & -0.460 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 2.285 & 0 \\ 0 & 2.010 \end{bmatrix}$$

$$U_2^T = \begin{bmatrix} -0.311 & -0.407 & -0.594 & -0.603 & -0.143 \\ 0.363 & 0.541 & 0.200 & -0.695 & -0.229 \end{bmatrix}$$

Latent Semantic Indexing

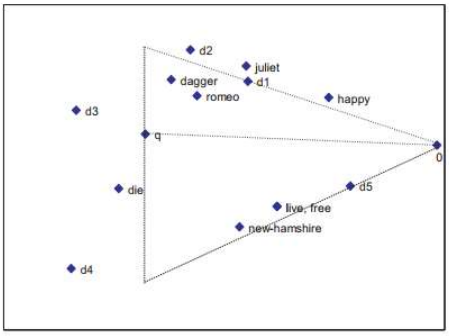


Figure 1: Geometric representation of documents $d_1, \ldots, d_5$, their terms, and query q.

- Maps terms and documents to same space in a reduced dimensionality
- Query can be mapped into same space
- Can identify similarity documents & terms

Issues with LSI (aka LSA)

- Linear model projecting to lower dimensional space (just like PCA)
- What would happen to word “mining”?
  - Pulling toward data mining
  - Pulling toward mineral mining
  - Mining can only occur once in projection
- Computational cost is main issue
  - 10K terms x 100K documents
- Ignores part of speech and position
- RankBrain (Google) has similar goal

Word2vec

Word2vec motivation

- Can we use a neural network to figure out relationships between words?
- Motivation:
  - If words are related, they must be nearby
  - Create a classifier to predict whether a target word is within context of given word

Example of context

"'Twas the night before Christmas and all"

Target word

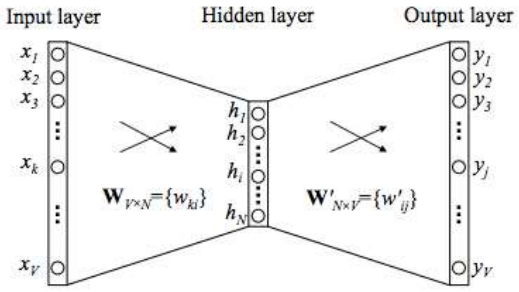
Defined a context window of +/- 2 words somewhat arbitrarily

Handling words…

- Neural networks like numbers
- How to put words into a neural net?
- What is the issue of doing this?
- [dog] = [0 0 0 0 0 0 0 0 1 0 0 0 0]
- 'cat' = [0 0 0 1 0 0 0 0 0 0 0 0 0]

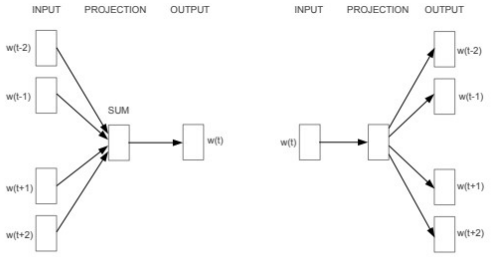
Toward an Embedding

- Supervised model:
  - Input & output are one-hot-encoding
- Hidden layer becomes the embedding
  - Vector representation of an input word
  - Similar in idea to an autoencoder



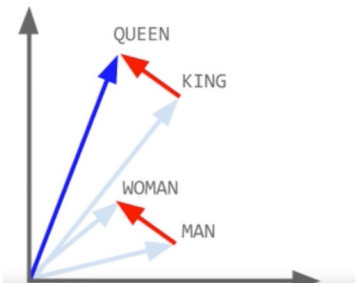
Word2vec Approaches: Mikolov 2013

- Skip-Gram
  - Input: Word
  - Predict: Neighbor of word
- CBOW
  - Continuous bag of words
  - Input: Bag of neighbors
  - Predict: Word



Word2vec

- Quantizes a word based on its context
- Similar words will have similar context
- Creates a meaningless ~200 len vector
  - Vectors can be easily compared
  - Cluster the data
  - Visualize the data
  - Search by synonym



https://medium.com/arvind-internet/applying-word2vec-on-our-catalog-data-2d74dfee419d

GloVe

- Global vectors for word representations
- Pennington et al. 2014
- Question is how related are two words

- Word2vec
  - Trained logistic regression actually
  - Ignored computing frequency of co-occurrence (was indirect in training)

- GloVe does gradient descent and includes frequency term

How to represent word meaning?

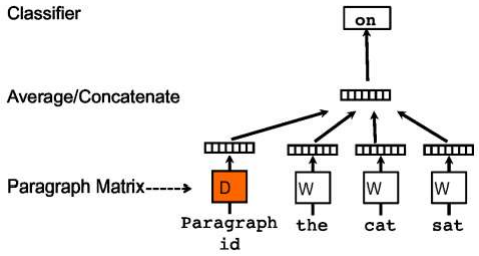
- Word2vec from Google
  - Distributed representations of words & phrases and their compositionality
  - Mikolov et al. 2013

- Take massive data corpus
- Use neural network to reduce space
- Train on word neighborhoods or combos
- Identify similarity of words in a vector

- Word → weighted vector of terms

Doc2Vec

- Same idea as word2vec except for a paragraph or document at a time
- Unsupervised way of learning set of words associated to a document
- Produces mapping for documents



Distributed Representations of Sentences & Documents
Quoc Le 2014

Latent Dirichlet Allocation

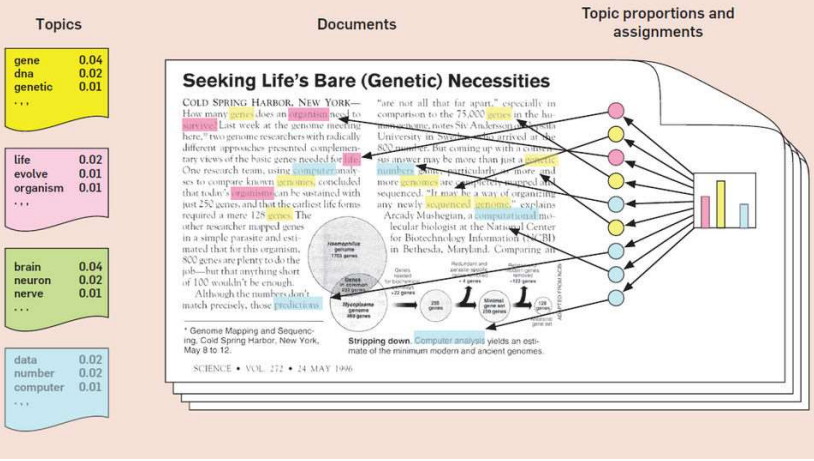
Goal of LDA

- Identify the topics across documents
- David Blei, Andrew Ng, Michael Jordan 2003
  http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf
- Given a corpus of documents:
  - Group them into separate topics
  - Figure out the features of each topic
- Clustering of words & documents
- LDA can be used for other purposes
  - Bioinformatics, image processing, etc.

Assumptions of LDA

- Topics are a human construct
- Machines approach topics as a set of co-occurring words
- Assumptions:
  - Documents are probability distributions of 'latent' topics
  - Topics are probability distributions of words

Topics ~ Words & Docs ~ Topics



http://vlsp.wikia.com/wiki/Latent_Dirichlet_Allocation?file=Latent-dirichlet-allocation.jpg

Generative Process

- Standard (non-generative) analyses:
  - Take a data set
  - Compute statistics that characterize it
  - Draw inferences
  - Data → Model
- Generative processes:
  - Start with a model
  - Create the data from the model
  - Model → Data

LDA's Generative Model

- To create a new document:
  - Determine the #words
  - Choose a mixture of topics (add to 100%)
  - Create words until have #words
    - Sampling a topic according to the multinomial mixture of topics
    - Sample a word based on topics' multinomial distribution
- No syntax, order but still topics
- Don't actually create documents but in fact work forward assuming this model

Conditional Probabilities Review

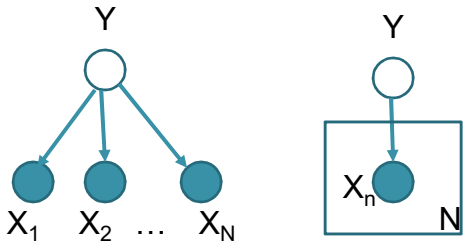
- P(A|B) = P(AB) / P(B)
- Probability of A given B
- B narrows the sampling space

- 52 cards in a deck
- Two red queens

- P(red queen) = 2/52
- P(red queen | red) = 2/26
- P(red queen | queen) = 2/4

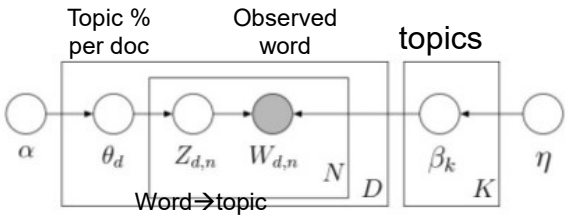
Theoretical Graphical Models

- Node is random variable
- Edges is conditional independence
- Observed variables are shaded
- Plates represent replicated structure



$$p(y, x_1 \dots x_N) = p(y) \prod_{i=1}^{N} p(x_i|y)$$

Graphical Model of LDA



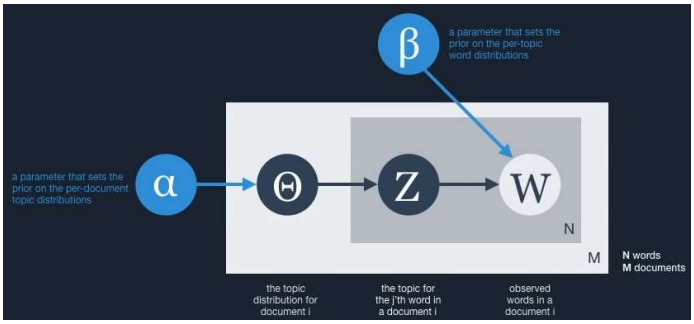
K – total number of topics
β_k – topic, a distribution over the vocabulary
D – total number of documents
Θ_d – per-document topic proportions
N – total number of words in a document (it fact, it should be N_d)
$Z_{d,n}$ – per-word topic assignment
$W_{d,n}$ – observed word
α, η – Dirichlet parameters

- Several **inference algorithms** are available (e.g. sampling based)
- A few **extensions** to LDA were created:
  - Bigram Topic Model

[D. Blei, **Probabilistic topic models**, *Communications of the ACM*, 2012]

- Two hyper-parameters α and η here
- Change notation & simplify next slide

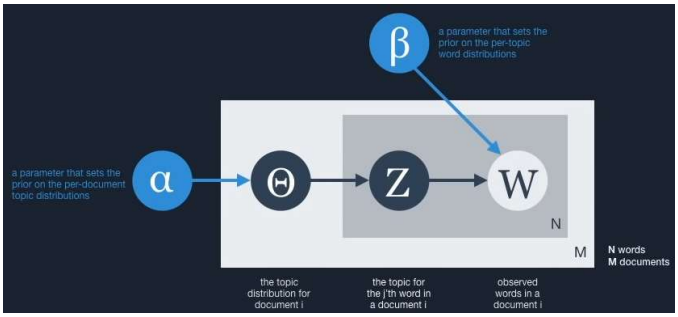
Latent Dirichlet Allocation Model



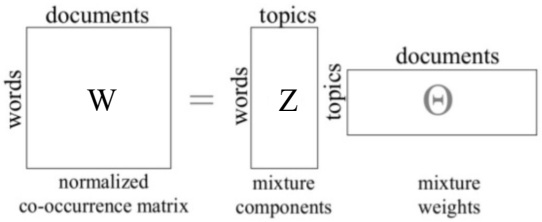
https://www.youtube.com/watch?v=3mHy4OSyRf0

- Two hyper-parameters:
  - α: per-document topic distribution
    - High: each document contains all topic
    - High: documents become more similar
  - β: per-topic word distribution
    - High: Each topic contains mixture of most words
    - High: Topics become more similar

Just have to solve this



$$p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta) = p(\theta \mid \alpha) \prod_{n=1}^{N} p(z_n \mid \theta) p(w_n \mid z_n, \beta)$$



Why Dirichlet?

- Two hyper-parameters
- Choosing distribution of:
  - Topics per document
  - Words per topic
- Hyper-parameters control prior distribution which is Dirichlet

PDF

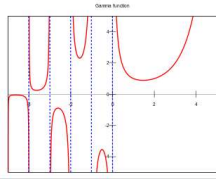
$$\frac{1}{\mathrm{B}(\boldsymbol{\alpha})}\prod_{i=1}^{K}x_i^{\alpha_i-1}$$

$$\text{where } \mathrm{B}(\boldsymbol{\alpha})=\frac{\prod_{i=1}^{K}\Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K}\alpha_i\right)}$$

$$\text{where } \boldsymbol{\alpha}=(\alpha_1,\ldots,\alpha_K)$$

wikipedia

$$\Gamma(z)=\int_0^\infty x^{z-1}e^{-x}\,dx$$



The gamma function along part of the real axis

Algorithm going backwards

- Start with a corpus of documents
- Select number of topics = K
- Randomly assign each word in each document to one of topics K
- For each document d & multiple epochs
  - For each word w in d
    - For each topic t
      - Compute PTD = P(topic t | document d)
        - % words assigned that topic in that doc
      - Compute PWT = P(word w | topic t)
        - % assignments to that topic for word w over corpus
    - Assign word w to new topic maximizing PTD*PWT
      - ~Probability of having a topic t generating word w

Quick example of update step

Document 1		Document 2		Document 3	
Eat	A	Cat	B	Cat	B
Fish	A	Dog	B	Eat	A
Vegetables	A	Pet	B	Fish	?
Fish	A	Pet	B	Cat	B
Eat	A	Fish	B	Fish	A

Example of topic distribution in documents

https://medium.com/ml2vec/topic-modeling-is-an-unsupervised-learning-approach-to-clustering-documents-to-discover-topics-fdfbf30e27d

- What topic for Fish in Doc 3?
  - P(topic t | document d)
    - P(A | Doc 3) = 2 A's of 4 known → 0.50
    - P(B | Doc 3) = 2 B's of 4 known → 0.50
  - P(word w | topic t)
    - P(Fish | A): 3As of 4 Fish → 0.75
    - P(Fish | B): 1B of 4 Fish → 0.25
  - Assign A: 0.50*0.75 > 0.50*0.25

Example Application

- Goal is to sort newly published articles into the right topics for scientists
- Train on 25 million PubMed abstracts
  - Find words, lemmatize, remove stop words
  - Create 100 topics maybe and run LDA
  - Identified word distributions of each topic
- Test on new document
  - 100 length vector (1 per topic)
  - Each tuple = fraction of words on that topic
- Can assign to highest topic (mixture)
- Can use vector to compare distributions
  - Use Jensen Shannon Distance

Advantages & Disadvantages

- Good:
  - Works well in identifying topics
  - Easy to understand and interpret
  - Number of uses beyond text
- Bad
  - Need to know # topics before you start
- Ugly
  - Each topic is independent (Dirichlet part)
    - Like saying all music fits only genre

Leveraging word2vec

What to do with word2vec output?

- Run a document → word2vec outputs
  - Did you include all the words?
  - Did you include the stop words?
  - What do you do with replicates?
- Doc A: 70 vectors each 200 long
- Doc B: 90 vectors each 200 long
- Goal is to:
  - Cluster
  - Classify
  - Predict
  - Model

Leveraging Word2Vec

- How can you compare 2 documents?

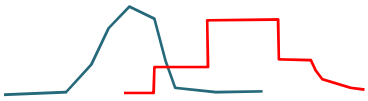
Leveraging Word2Vec

- Centroid of the words per document
- Compare centroids using distance

- Other idea: Compare words somehow

General Idea

- What is the distance between them?



- What is dist(Cambridge, New Haven)?

- What is distance of MA to CT?

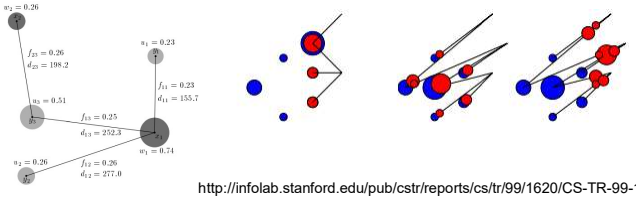
https://wikitravel.org/en/New_England

Image similarity using EMD

- Image retrieval system base on EMD similarity measure and S-tree
  - Le & Van 2015

Earth Mover Distance (EMD)

- Distance between two distributions
- Proportional to min “work” to change one distribution to the other
  - Think of moving each chunk from A to B
  - Flows from heavy to lighter
- EMD(x,y) = min Work(F,x,y)/min(wS,uS)

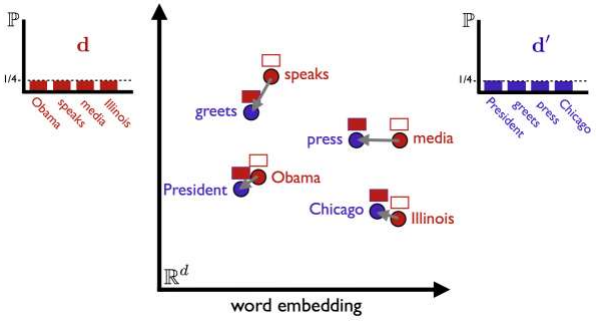


http://infolab.stanford.edu/pub/cstr/reports/cs/tr/99/1620/CS-TR-99-1620.ch4.pdf

Word Mover Distance

- From Word Embeddings to Document Distances. Kusner Sun Kolkin Weinberger 2015

- "Obama speaks to the media in Illinois"
- "President greets the press in Chicago"



https://markroxor.github.io/gensim/static/notebooks/WMD_tutorial.html

Optimization Problem

- Related to the transportation problem
- What is min cost to move goods from a supplier to a demander where goods have a known size and location

- Solution uses simplex algorithm
  - Constrained optimization problem
  - Standard approach used in many fields

Word Mover Distance WMD

- Remove stop words
- Use word2vec vectors so word i → vec x_i
- Distance of words = cost = c(i,j) = $||x_i-x_j||^2$
- Flow matrix T where
  - T_{ij} = flow from d_i to d_j so T_{src_dest}
  - T_{ij} >= 0
- d_k set to 1/#words for each sentence
  - Dividing up word to flow out $\Sigma_j T_{ij} = d_i$
  - Input to word has to add up $\Sigma_i T_{ij} = d_j$
- Distance of sentences = $\Sigma_{i,j} T_{ij}c(i,j)$
  - Min cost to move all words from d_i to d_j

Example of distances

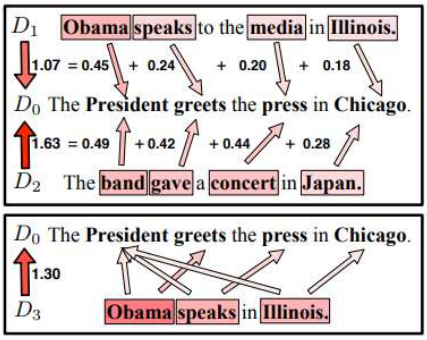


Figure 2. (Top:) The components of the WMD metric between a query D_0 and two sentences D_1, D_2 (with equal BOW distance). The arrows represent flow between two words and are labeled with their distance contribution. *(Bottom:)* The flow between two sentences D_3 and D_0 with different numbers of words. This mismatch causes the WMD to move words to multiple similar words.

- Top: D1 closer to D0 than D2
- Bottom: D3 (far) has 3 words D1 has 4 (close)

Kusner et al. WMD paper

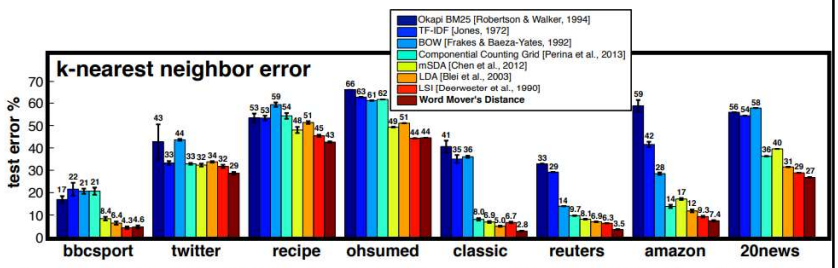


Figure 3. The kNN test error results on 8 document classification data sets, compared to canonical and state-of-the-art baselines methods.

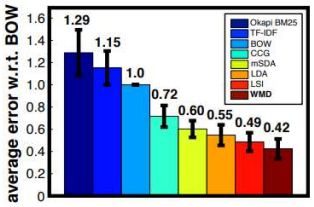


Figure 4. The kNN test errors of various document metrics averaged over all eight datasets, relative to kNN with BOW.

Word Mover Distance

- Good
  - Hyper-parameter free
  - Interpretable
  - Implemented in Gensim's WmdSimilarity
  - Currently gives best performance
- Bad
  - Slow but has lower bounds
    - Centroid distance = one of lower bounds
- Ugly

Compound Words

Named Entity Extraction

- New England Patriots
- Bank of America
- Staples
- May want to know if person/place/thing

- Deoxyribonucleic acid
- Core i7 processor
- Parking lot

日本語の表記においては，漢字や仮名だけでなく，ローマ字やアラビア数字，さらに句読点や括弧類などの記述記号を用いる。これらを組み合わせて表す日本語の文書では，表記上における種々の問題がある。

特朗普
川普

Collocations

- Statistical approaches to the problem
  - https://nlp.stanford.edu/fsnlp/promo/colloc.pdf
- Hypothesis test:
  - Statistically evaluate whether adjacent words are part of a compound noun
  - Null hypothesis: not related
  - T-test across a large corpus
- Exploring statistical analysis based on the co-occurrence frequencies

Compound Words

Links

- https://www.youtube.com/watch?v=-2NslwwHCbU
- https://shangjingbo1226.github.io/2017-08-11-kdd-tutorial/

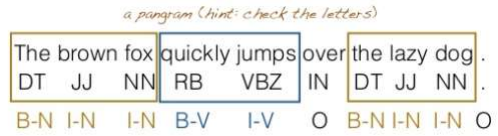
Statistics measuring phrase quality:

- Frequency: # occurrences in large corpus
- Concordance: frequency > random
- Informative: “an algorithm”
- Completeness: “Support Vector” vs. SVM

Goal: multilingual & no manual effort

Two supervised methods

- Parsing (or partial parsing)
  - Leverage the grammar
  - Identify the noun phrases
- Chunking
  - Use POS info to identify noun phrases



https://www.slideshare.net/asdkfjqlwef/text-mining-55-information-extraction

Problem: require human training

ToPMine 2015 El-Kishky et al.

- Extracts quality phrases statistically
- Does not require training data
- Use phrase mining, then topic modeling

- Find frequent contiguous words
- Perform 'rectified' frequency
  - Remove CNN from "NN" and "Conv N"

Phrase	Original Frequency	Rectified Frequency
Convolutional neural network	75	70
Neural network	200	130
Convolutional neural	80	10

ToPMine

- Frequency: # occurrences in large corpus
- Concordance: frequency > random
  - Chisq test: $\chi^2 = \sum \frac{(O-E)^2}{E}$ or PMI(x,y) = log$\frac{p(x,y)}{p(x)p(y)}$
  - Compare #stdev from expected and iteratively merge words
  - Perform hierarchical clustering with a threshold essentially
- Informative: "an algorithm"
- Completeness: "Support Vector" vs. SVM

Results from ToPMine paper

	Topic 1	*Topic 2*	*Topic 3*	*Topic 4*	*Topic 5*
1-grams	problem	word	data	programming	data
	algorithm	language	method	language	patterns
	optimal	text	algorithm	code	mining
	solution	speech	learning	type	rules
	search	system	clustering	object	set
	solve	recognition	classification	implementation	event
	constraints	character	based	system	time
	programming	translation	features	compiler	association
	heuristic	sentences	proposed	java	stream
	genetic	grammar	classifier	data	large
n-grams	genetic algorithm	natural language	data sets	programming language	data mining
	optimization problem	speech recognition	support vector machine	source code	data sets
	solve this problem	language model	learning algorithm	object oriented	data streams
	optimal solution	natural language processing	machine learning	type system	association rules
	evolutionary algorithm	machine translation	feature selection	data structure	data collection
	local search	recognition system	paper we propose	program execution	time series
	search space	context free grammars	clustering algorithm	run time	data analysis
	optimization algorithm	sign language	decision tree	code generation	mining algorithms
	search algorithm	recognition rate	proposed method	object oriented programming	spatio temporal
	objective function	character recognition	training data	java programs	frequent itemsets

Table 4: Five topics from a 50-topic run of ToPMine framework on our full DBLP abstracts dataset. Overall we see coherent topics and high-quality topical phrases we interpret as search/optimization, NLP, machine learning, programming languages, and data mining

SeqPhrase & AutoPhrase

- Automatic extraction of phrases from corpora within specific contexts
- If know domain, then can provide some training data or perhaps Wikipedia knowledge

KDD Pattern Mining

KDD

- Knowledge Discovery from Data Bases
- Knowledge Discovery from Data
- Knowledge Discovery & Data Mining

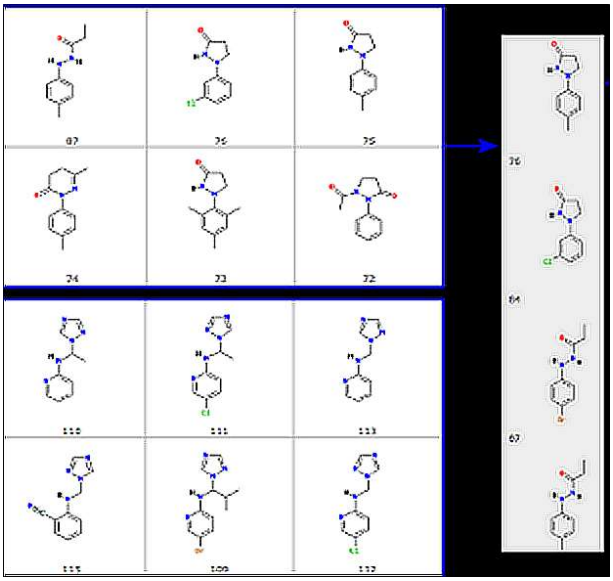
Pattern Discovery

- Looking at transactional or other data
- Pattern = set of item, part of sequence or structure that co-occur
- Finding patterns that look interesting
- Frequently occurring patterns
- Irregularities

Related areas

- Reasoning:
  - Associations or causality
- Data miners
- Classification & Clustering
  - Patterns are part of the vocabulary
- Marketing
- Biological and other research

Finding Frequent Subgraphs



- http://chemmine.ucr.edu/help/

Market Basket Analysis

- What do customers buy together?
- Milk and bread
- iphone, cover, protector, charger
- Urban legend of diapers and beer
  - Theory is nostalgic young men buy beer with diapers
  - 1992 analysis of Osco drug stores—no age or gender was included in the analysis but there was a pattern
  - http://www.theregister.co.uk/2006/08/15/beer_diapers/

Association Rules

- People who buy an iphone, buy a case
- iphone → case

- http://www.speckproducts.com/apple/iphone-cases/iphone-6-cases/candyshell-grip-iphone-6-cases/IP6-CS-GRIP.html#start=6
- What statistics or information would you want to believe this is a meaningful rule?

Association Rules

- iphone → case
- Support =
  - Percentage of all transactions where rule holds (Best Buy: 2%)
  - P(iphone U case)
- Confidence =
  - Percentage of customers who bought an iphone bought a case (Best Buy: 80%)
  - P(case | iphone) = Support(case,iphone)/Support(iphone)

Vocabulary

- Itemset = set of items (in cart)
- Frequency of itemset
  - #transactions that include the itemset
  - Also called “count”
- Minimum support = % of transactions that contain a given item[set]
- Frequent itemset is an itemset that exceeds the minimum support

Mining Frequent Itemsets

- Confidence
  - = P(case | iphone)
  - = Support(case,iphone) / Support(iphone)
- Process
  - Count the frequencies of each item set
  - Compute the confidence
  - Identify “strong” association rules defined as:
    - Satisfy a minimum support
    - Satisfy a minimum confidence
- Why would this be difficult?

Challenge of mining itemsets

- $\binom{a}{b} = \frac{a!}{b!(a-b)!}$ = a choose b
- For 20 items you have
  - $\binom{20}{1}$ 1-itemsets → 20! / (19! * 1!) = 20
  - $\binom{20}{2}$ 2-itemsets → 20! / (18! * 2!) = 190
  - $\binom{20}{3}$ 3-itemsets → 20! / (17! * 3!) = 1140
  - …
  - $\binom{20}{20}$ 20 itemsets → 20! / (20! * 0!) = 1
- Total = 1,048,575 itemsets

Frequent Itemsets

- For 20 items you have
- Total = 1,048,575 itemsets

- How many items are in a supermarket?

Frequent Itemsets

- For 20 items you have
- Total = 1,048,575 itemsets

- How many items are in a supermarket?
  - 42,214 items
  - http://www.fmi.org/research-resources/supermarket-facts

Closed Frequent Itemset

- Itemset X is closed in data set D if there is no proper super-itemset Y such that Y^S has the same support count as X
- Closed frequent itemset = closed and frequent
- Max frequent itemset = itemset that is frequent and has no super-itemset Y where Y is frequent

Example

- http://stats.stackexchange.com/questions/77465/maximal-closed-frequent-answer-included
- Let MinSupport = 50%
  - % of itemsets containing the subset of interest
- MinSupportCount = 3
  - 6 item sets and 50% of 6 = 3
- A=4 B=2 C=5 D=4 E=6
  - Which are frequent?

My dataset:
1 : A, B, C, E
2 : A, C, D, E
3 : B, C, E
4 : A, C, D, E
5 : C, D, E
6 : A, D, E

Example

- http://stats.stackexchange.com/questions/77465/maximal-closed-frequent-answer-included
- MinSupport = 50%
  - % of item sets containing the subset of interest
- MinSupportCount = 3
  - 6 item sets and 50% of 6 = 3
- A=4 B=2 C=5 D=4 E=6
  - Which are frequent?

$$
\begin{array}{l}
My\ dataset: \\
1: A, B, C, E \\
2: A, C, D, E \\
3: \quad B, C, E \\
4: A, C, D, E \\
5: \quad C, D, E \\
6: \quad A, D, E
\end{array}
$$

- AB=1 AC=3 AD=3 AE=4
- BC=2 BD=0 BE=2
- CD=3 CE=5
- DE=4

Example

- A=4 C=5 D=4 E=6
- AB=1 AC=3 AD=3 AE=4
- BC=2 BD=0 BE=2
- CD=3 CE=5
- DE=4

$$
\begin{array}{l}
My\ dataset: \\
1: A, B, C, E \\
2: A, C, D, E \\
3: \quad B, C, E \\
4: A, C, D, E \\
5: \quad C, D, E \\
6: \quad A, D, E
\end{array}
$$

- Closed = no super-itemset Y that has the same support count as X
- Let X = A, A has count=4
- Is there a super-itemset (A+?) that also has count=4

Example

- A=4 C=5 D=4 E=6
- AB=1 AC=3 AD=3 AE=4
- BC=2 BD=0 BE=2
- CD=3 CE=5
- DE=4

$$
\begin{array}{l}
\textit{My dataset:} \\
1: A, B, C, E \\
2: A, C, D, E \\
3: \quad B, C, E \\
4: A, C, D, E \\
5: \quad C, D, E \\
6: \quad A, D, E
\end{array}
$$

- Closed = no super-itemset Y that has exactly the same support count as X

- Let X = C, C has count=5
- Is there a super-itemset (C+?) that also has count=5?
- Looking at top row, which can we ignore?

Example

- A=4 C=5 D=4 E=6
- AB=1 AC=3 AD=3 AE=4
- BC=2 BD=0 BE=2
- CD=3 CE=5
- DE=4

$$
\begin{array}{l}
\textit{My dataset:} \\
1: A, B, C, E \\
2: A, C, D, E \\
3: \quad B, C, E \\
4: A, C, D, E \\
5: \quad C, D, E \\
6: \quad A, D, E
\end{array}
$$

- Closed = no super-itemset Y that has exactly the same support count as X

- Let X = D, D has 4
- Is there a super-itemset (D+?) that also has count=4?

Example

- A=4 C=5 D=4 E=6
- AB=1 AC=3 AD=3 AE=4
- BC=2 BD=0 BE=2
- CD=3 CE=5
- DE=4

$$
\begin{array}{l}
\textit{My dataset:} \\
1: A, B, C, E \\
2: A, C, D, E \\
3: \quad B, C, E \\
4: A, C, D, E \\
5: \quad C, D, E \\
6: \quad A, D, E
\end{array}
$$

- Max frequent itemset = frequent and has no super-itemset Y where Y is frequent
- Let X = D, D has 4
- Is there a super-itemset (D+?) that is frequent?

Example

- A=4 C=5 D=4 E=6
- AB=1 AC=3 AD=3 AE=4
- BC=2 BD=0 BE=2
- CD=3 CE=5
- DE=4

$$
\begin{array}{l}
\textit{My dataset:} \\
1: A, B, C, E \\
2: A, C, D, E \\
3: \quad B, C, E \\
4: A, C, D, E \\
5: \quad C, D, E \\
6: \quad A, D, E
\end{array}
$$

- ABC=1 ABD=0 ABE=1 ACD=2
- ACE=3 ADE=3
- BCD=0 BCE=2 CDE=3
- Is CD closed? Maximal?
- Is CE closed? Maximal?

Example

- A=4 C=5 D=4 E=6
- AB=1 AC=3 AD=3 AE=4
- BC=2 BD=0 BE=2
- CD=3 CE=5
- DE=4

$My\ dataset:$
$1: A, B, C, E$
$2: A, C, D, E$
$3: B, C, E$
$4: A, C, D, E$
$5: C, D, E$
$6: A, D, E$

- ABC=1 ABD=0 ABE=1 ACD=2
- ACE=3 ADE=3
- BCD=0 BCE=2 CDE=3

- Is CD closed? Maximal?
- Is CE closed? Maximal?

Example

- A=4 C=5 D=4 E=6
- AB=1 AC=3 AD=3 AE=4
- BC=2 BD=0 BE=2
- CD=3 CE=5
- DE=4

$My\ dataset:$
$1: A, B, C, E$
$2: A, C, D, E$
$3: B, C, E$
$4: A, C, D, E$
$5: C, D, E$
$6: A, D, E$

- ABC=1 ABD=0 ABE=1 ACD=2
- ACE=3 ADE=3
- BCD=0 BCE=2 CDE=3

- Is CD closed? No Maximal? No

Example

- A=4 C=5 D=4 E=6
- AB=1 AC=3 AD=3 AE=4
- BC=2 BD=0 BE=2
- CD=3 CE=5
- DE=4

$$
\begin{array}{l}
\textit{My dataset}: \\
1: A, B, C, E \\
2: A, C, D, E \\
3: \quad B, C, E \\
4: A, C, D, E \\
5: \quad C, D, E \\
6: \quad A, D, E
\end{array}
$$

- ABC=1 ABD=0 ABE=1 ACD=2
- ACE=3 ADE=3
- BCD=0 BCE=2 CDE=3

- Is AD closed? Maximal?

Example

- A=4 C=5 D=4 E=6
- AB=1 AC=3 AD=3 AE=4
- BC=2 BD=0 BE=2
- CD=3 CE=5
- DE=4

$$
\begin{array}{l}
\textit{My dataset}: \\
1: A, B, C, E \\
2: A, C, D, E \\
3: \quad B, C, E \\
4: A, C, D, E \\
5: \quad C, D, E \\
6: \quad A, D, E
\end{array}
$$

- ABC=1 ABD=0 ABE=1 ACD=2
- ACE=3 ADE=3
- BCD=0 BCE=2 CDE=3

- Is AD closed? No Maximal? No

Example

- A=4 C=5 D=4 E=6
- AB=1 AC=3 AD=3 AE=4
- BC=2 BD=0 BE=2
- CD=3 CE=5
- DE=4

$$
\begin{array}{l}
\textit{My dataset}: \\
1: A, B, C, E \\
2: A, C, D, E \\
3: \quad B, C, E \\
4: A, C, D, E \\
5: \quad C, D, E \\
6: \quad A, D, E
\end{array}
$$

- ABC=1 ABD=0 ABE=1 ACD=2
- ACE=3 ADE=3
- BCD=0 BCE=2 CDE=3

- Is CE closed? Maximal?

Example

- A=4 C=5 D=4 E=6
- AB=1 AC=3 AD=3 AE=4
- BC=2 BD=0 BE=2
- CD=3 CE=5
- DE=4

$$
\begin{array}{l}
\textit{My dataset}: \\
1: A, B, C, E \\
2: A, C, D, E \\
3: \quad B, C, E \\
4: A, C, D, E \\
5: \quad C, D, E \\
6: \quad A, D, E
\end{array}
$$

- ABC=1 ABD=0 ABE=1 ACD=2
- ACE=3 ADE=3
- BCD=0 BCE=2 CDE=3

- Is CE closed? Yes Maximal? No

Why have closed pattern idea?

- If you have many frequent patterns perhaps with low support
  - Computationally impossible
  - Exceed memory and computation
- Max frequent is a different summary
  - Lose information
- "Closed" is a summary of patterns
  - Don't lose information
  - More widely used

Apriori Algorithm

- Standard for finding frequent itemsets
- Agrawal and Srikant 1994
- Boolean association rules (iphone→case)
- Identify all item sets of with count 1= L1
- Identify all item sets of with count i = Li

How Apriori Works

- Scan through database to find each individual item → L1 set of 1-items
  - Basically every individual item
- Scan through database to try to grow the L1 members into L2 of 2-item combos
- Rinse repeat → Lk with each level requiring a single scan through the database until no more patterns
- Return all frequent patterns

Apriori Property

- Subsets of frequent itemset must also be frequent itemsets
- ABC occurs 7 times so each of following must also occur >= 7 times
  - A, B, C, AB, BC, AC
- Antimonotonicity:
  - Class of properties where if a set fails, all supersets fail as well
  - If AB < 5, then ABC<5, ABD<5, AB* < 5

Apriori Property

- Subsets of frequent itemset must also be frequent itemsets
- Is the converse true?
- If ABC is not frequent (i.e. below threshold), can any of its subsets be frequent subsets?

Join Step

- Take two itemsets with k entries each
- Assume all itemsets are sorted
- Join them if the first k-1 entries match and the last two entries are different
  - Why? Because it ensures no duplicates
- A,C,D,E,K
- A,C,D,E,F
- → A,C,D,E,F,K

Prune step

- Result of join step is a set of k-itemsets some of which may be frequent
- Which ones are frequent?
  - Could go through the dataset and count
  - Might be millions of entries
- Use earlier property that for k-itemset to be frequent it cannot contain any k-1 itemset that is not frequent
  - Maintain a hash tree of all frequent itemsets

Example: Define Frequent as ≥ 2

Itemset
A,B,E
B,D
B,C
A,B,D
A,C
B,C
A,C
A,B,C,E
A,B,C

A: 6
B: 7
C: 6
D: 2
E: 2

A,B: ?
A,C: ?
A,D: ?
A,E: ?
B,C: 4
B,D: 2
B,E: 2
C,D: 0
C,E: 1
D,E: 0

Grow the 3-set

Example: Define Frequent as ≥ 2

Itemset
A,B,E
B,D
B,C
A,B,D
A,C
B,C
A,C
A,B,C,E
A,B,C

A: 6
B: 7
C: 6
D: 2
E: 2

A,B: 4
A,C: 4
A,D: 1
A,E: 2
B,C: 4
B,D: 2
B,E: 2
C,D: 0
C,E: 1
D,E: 0

Grow the 3-set

Example: Define Frequent as ≥ 2

Itemset
A,B,E
B,D
B,C
A,B,D
A,C
B,C
A,C
A,B,C,E
A,B,C

A: 6
B: 7
C: 6

A,B: 4
A,C: 4
A,D: 1

Join them if the first k-1 entries match and the last two entries are different

B,E: 2
C,D: 0
C,E: 1
D,E: 0

Grow the 3-set

Example: Define Frequent as ≥ 2

Itemset
A,B,E
B,D
B,C
A,B,D
A,C
B,C
A,C
A,B,C,E
A,B,C

A: 6
B: 7
C: 6
D: 2
E: 2

A,B: 4
A,C: 4
A,D: 1
A,E: 2
B,C: 4
B,D: 2
B,E: 2
C,D: 0
C,E: 1
D,E: 0

A,B,C: 2
A,B,E: 2
A,C,E:
B,C,D:
B,C,E:
B,D,E:

Check our assumption of join need only consider match first k-1

1) A,B: 4
2) A,C: 4
3) A,E: 2
4) B,C: 4
5) B,D: 2
6) B,E: 2

Non-freq
A,D: 1
C,D: 0
C,E: 1
D,E: 0

1+2 A,B,C
1+3 A,B,E
1+5 A,B,D
2+3 A,C,E
2+5 ~~A,B,C,D~~
2+6 ~~A,B,C,E~~
4+5 B,C,D
4+6 B,C,E
5+6 B,D,E

Others
1+4 A,B,C
1+6 A,B,E
2+4 A,B,C
3+4 A,B,C,E
3+6 A,B,D,E
Found all 3-sets in this example already

The Apriori Algorithm—An Example