Text mining methods:
Topic modelling & Graph-based keywords extraction

Jelena Jovanović

Email: jeljov@gmail.com

Web: http://jelenajovanovic.net

OVERVIEW

- Topic modelling methods
  - LDA
- Graph-based methods
  - KeyGraph
  - TextRank

TOPIC MODELLING

TOPIC MODELLING METHODS

Topic modeling methods are statistical methods that analyze the words of the given collection of documents to

- discover the underlying themes,
- how those themes are connected to each other, and
- how they change over time

Latent Dirichlet allocation (LDA)

Latent Dirichlet allocation (LDA) is cited as the simplest topic modelling method

LDA assumptions:

- Topic is a *distribution over a fixed vocabulary*
- There is a fix set of topics for a collection of documents
- Each document in a collection has its own distribution over the given (fixed) set of topics
  - as a consequence, each document exhibits multiple topics

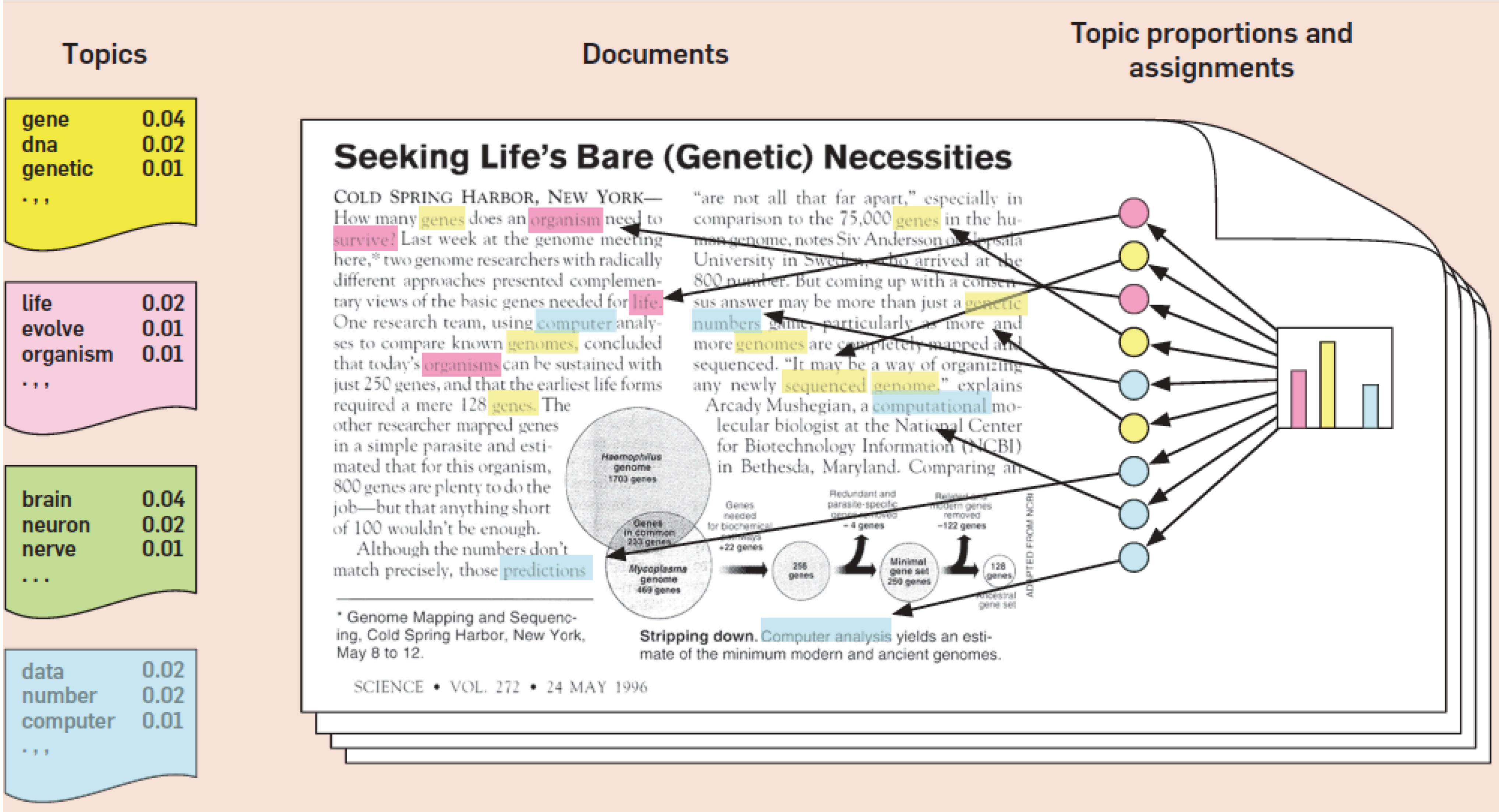
LDA'S GENERATIVE PROCESS

First, specify a set of topics for the given documents collection

Then, for each document in the collection, we generate words in a two-stage process:

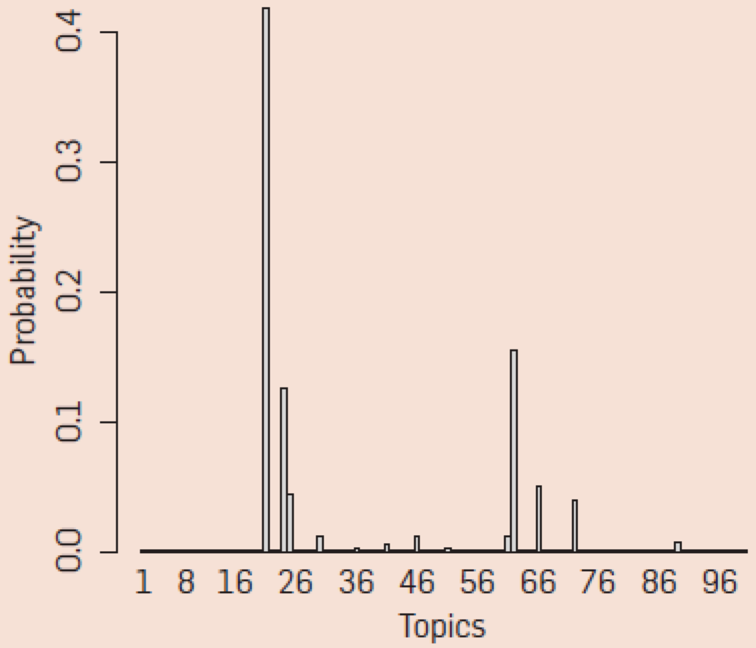
1) Randomly choose a distribution over topics
2) For each word (to be created) in the document
   a) Randomly choose a topic from the distribution over topics in step #1
   b) Randomly choose a word from the selected topic, that is, the corresponding distribution over the vocabulary

LDA'S GENERATIVE PROCESS



Source: Blei, David. 2012. Probabilistic topic models. Communications of the ACM 55(4):77–84.

LDA RESULTS



"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

Real results for the previous example article, obtained by fitting a 100-topic LDA model over 17,000 articles from the Science journal

LDA – THE NAME ORIGIN

- **Dirichlet** comes from the name of the distribution (Dirichlet dist.) that is used to draw the per-document topic distribution
- **Latent** comes from the fact that topics (their distribution and structure) are *hidden*, *unobservable*, and have to be inferred / mined from the observable items (words)

INTERPRETATION OF LDA INFERRED TOPICS

- Topics inferred by LDA are not always easily interpretable by humans
- Several attempts at facilitating the task of topic interpretation
- Examples:
  - Interactive visualization of LDA results (topics, terms) and documents, such as this Wikipedia browser
  - Using alternative measures for ranking terms within a topic, e.g.
    - *Lift* - the ratio of a term's probability within a topic to its marginal probability across the corpus
    - *Pointwise Mutual Information (PMI)* – combines frequency ranking and ranking based on co-occurrence of the frequent terms

INTERPRETATION OF LDA INFERRED TOPICS

- LDAVis:
  - URL: https://github.com/cpsievert/LDAvis
  - Combines interactive visualization and alternative ways of term ranking
  - Introduces the measure of term *relevance*:

$$r(w,k|\lambda) = \lambda * \log(\phi_{kw}) + (1-\lambda) * \log\left(\frac{\phi_{kw}}{p_w}\right)$$

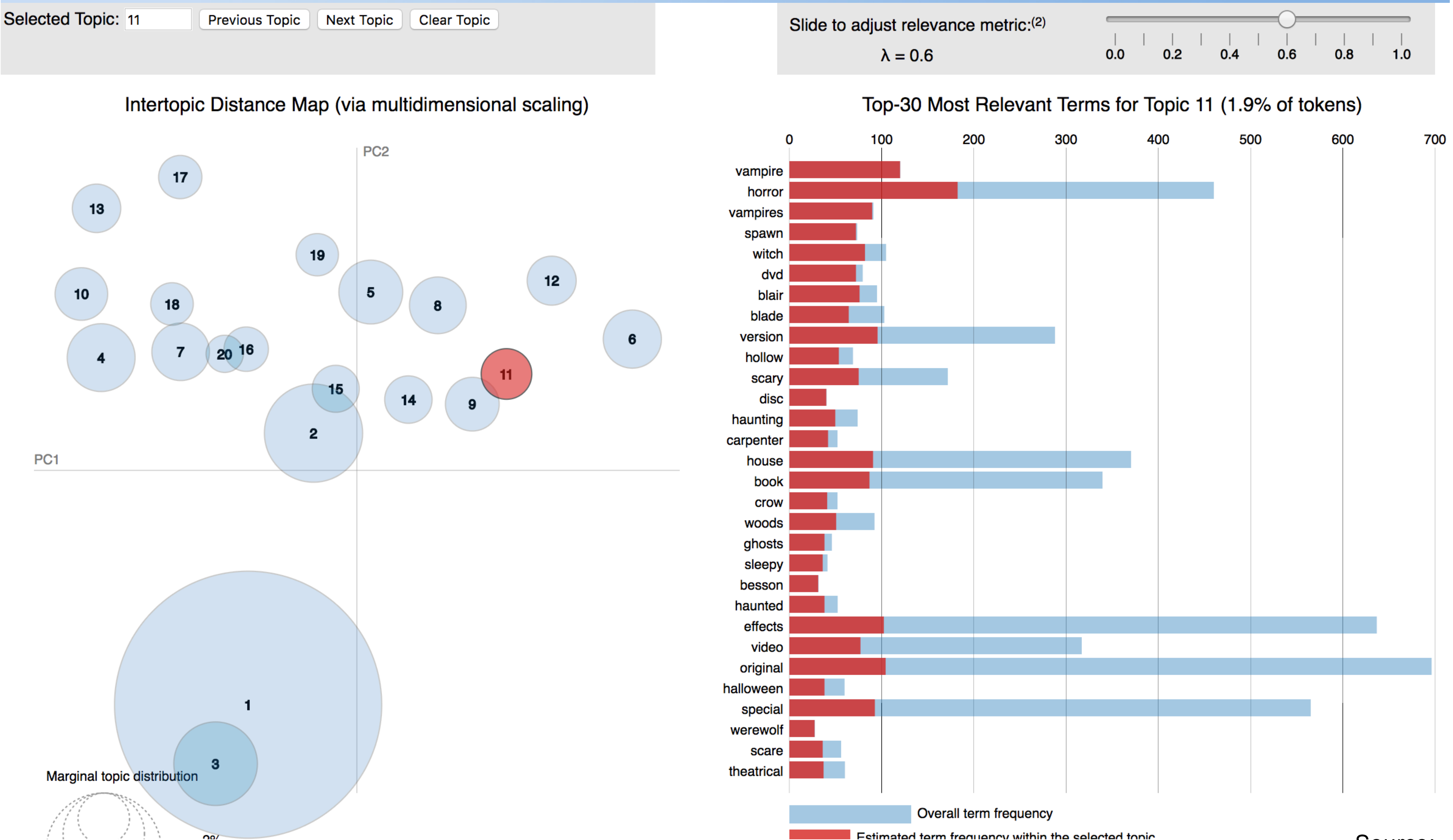
ϕ_{kw} - probability of the term w in the topic k

p_w - probability of the term w in the overall corpus (marginal prob.)

λ - the parameter (0-1); the authors' study found 0.6 to be the best value

Sievert, C. & Shirley, K. (2014). LDAvis: A method for visualizing and interpreting topics. Proc. of the Workshop on Interactive Language Learning, Visualization, and Interfaces. URL: http://nlp.stanford.edu/events/illvi2014/papers/sievert-illvi2014.pdf

LDAvis EXAMPLE



Source:
http://cpsievert.github.io/LDAvis/reviews/vis/

Check this short talk on LDAVis:
https://speakerdeck.com/bmabey/visualizing-topic-models

LDA ASSUMPTIONS (RESTRICTIONS)

The assumptions that LDA makes:

- bag of words assumption: the order of words in a document does not matter
- the order of documents (in the corpus) does not matter
- the number of topics is assumed to be known and is fixed
- topics are mutually unrelated

Other, more complex topic modelling methods relax these assumptions

TOPIC MODELS BEYOND LDA

- *Dynamic topic model* respects the ordering of the documents in a collection
- *Correlated topic model* allows the occurrence of topics to exhibit correlation
- *Spherical topic model* allows words to be unlikely in a topic
- *Structural topic model* includes document metadata as covariates that might affect
  - topical prevalence - how much a document is associated with a topic
  - topical content – the words used within a topic

SOFTWARE LIBRARIES FOR TOPIC MODELLING

- A variety of options in R:
  - lda: https://cran.r-project.org/package=lda
  - topicmodels: https://cran.r-project.org/package=topicmodels
  - stm: http://www.structuraltopicmodel.com/
- Also, several Python libraries:
  - Gensim: https://radimrehurek.com/gensim/
  - lda: http://pythonhosted.org//lda/
- In Java:
  - MALLET Topic Modelling lib: http://mallet.cs.umass.edu/topics.php

GRAPH-BASED METHODS: KEYGRAPH

H. Sayyadi, L. Raschid. "A Graph Analytical Approach for Topic Detection", ACM Transactions on Internet Technology (TOIT), 2013

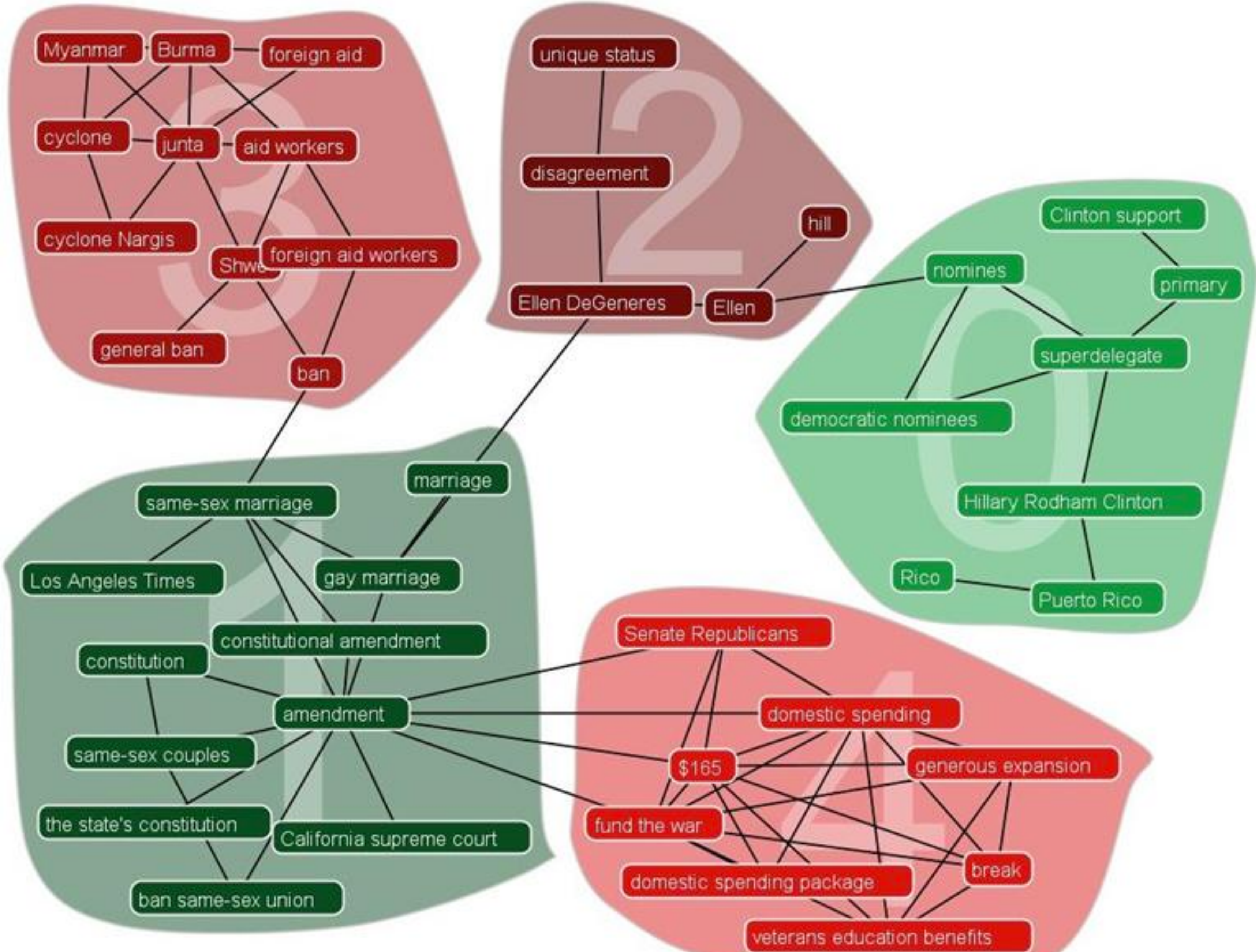
KeyGraph in a Nutshell

- Represents a collection of documents as a keyword co-occurrence graph
- Uses an off-the shelf community detection algorithm to group highly co-occurring keywords into “communities” (clusters)
- The detected communities prove to be good proxies for document topics

KEYGRAPH: THE INTUITION

- Keywords co-occur when there is a meaningful topical relationship between them
- Making an analogy to real-world social networks - where people connect if they share a common ‘topic’ (interest, activity, affiliation, etc.) - KeyGraph is modelled as a social network of keywords

ILLUSTRATION OF KEYGRAPH RESULT



Source: http://keygraph.codeplex.com/

KEYGRAPH ALGORITHM

1) Build a keywords co-occurrence graph for the given document collection
2) Community detection and extraction of topic features
3) Assigning topics to documents (based on the detected topic features)
4) Merging topics with significant document overlap

KEYGRAPH ALGORITHM: STEP 1

- Create the initial keywords co-occurrence graph
  - nodes are keywords (nouns, noun phrases, named entities) extracted from the corpus
  - an edge is established between two nodes if the corresponding keywords co-occur in at least one document;
  - edges are weighted by the count of the co-occurrences
- The initial graph is filtered based on
  - the document frequency (df) of individual keywords
  - the probability of co-occurrence of each pair of keywords

$$p(k_i|k_j) = \frac{df_{i\cap j}}{df_j} \quad ; \quad p(k_j|k_i) = \frac{df_{i\cap j}}{df_i}$$

KeyGraph Algorithm: Step 2

- Community detection
  - relies on an off-the shelf algorithm for community detection (relational clustering) based on the *edge betweenness centrality (Bc)* metric
  - *Bc* for an edge is defined as the count of the shortest paths, for all pairs of nodes in the network, that pass through that edge
  - in an iterative process, all edges with high *Bc* are removed, thus cutting all inter-community connections and splitting the graph into several components, each corresponding to one (topical) community
- Extraction of topic features
  - the highly co-occurring keywords in each component of the KeyGraph form the features for the corresponding topic

KEYGRAPH ALGORITHM: STEP 3

- Each community of keywords forms a *feature document* f_t, for the corresponding topic t
- The likelihood of the topic t for a document *d* is determined as the cosine similarity of *d* and the feature document ft :

$$p(t|d) = \frac{cosine(d, f_t)}{\sum_{t \in T} cosine(d, f_t)}$$

- Each document can be associated with multiple topics (each with a different likelihood)

KEYGRAPH ALGORITHM: STEP 4

- If case multiple documents are assigned to a pair of topics, it is assumed that those two topics are sub-topics of the same parent topic, and they are merged
- The allowed level of overlap between any two topics is controlled by a parameter (threshold)

ADVANTAGES OF THE KEYGRAPH METHOD

- Comparable performance (precision, recall, F1) to state of the art topic modelling methods
- Capable of filtering noisy irrelevant (social media) posts, thus creating smaller clusters of relevant documents for each topic
- Its running time is linear in the size of the document collection
  - it significantly outruns LDA method on large datasets (>50,000 documents)
- It is robust with respect to the parameters, that is, its performance does not vary much with the change in parameter values

FIND MORE ABOUT KEYGRAPH

- Implementation in Java and further information available at: https://keygraph.codeplex.com/

GRAPH-BASED METHODS: TEXTRANK

Mihalcea, R. & Tarau, P. (2004). TextRank: Bringing order into texts. In D. Lin & D. Wu (Eds.), Proc. of Empirical Methods in Natural Language Processing (EMNLP) 2004 (pp. 404–411), Barcelona, Spain, July. Association for Computational Linguistics.

GRAPH-BASED RANKING METHODS

- TextRank is a *graph-based ranking method*
- The basic idea behind such methods is that of ‘voting’ or ‘recommendation’:
  - when node A links to the node B, it is basically casting a vote for B
  - the higher the number of votes a node receives, the higher is its importance (in the graph)
  - the importance of the node casting the vote (A) determines how important the vote itself is

TEXTRANK METHOD

- It is based on the Google's original PageRank model for computing a node's importance score:

$$S(N_i) = (1 - d) + d * \sum_{j \in In(N_i)} \frac{1}{|Out(N_j)|} S(N_j)$$

$S(N_i)$ – score for node i

$Out(N_i)$ – the set of nodes that node N_i points to

$In(N_i)$ – the set of nodes that point to N_i

d – the prob. of going from N_i to one of $Out(N_i)$ nodes; 1-d is the prob. of jumping to a random node in the graph (the random surfer model)

TEXTRANK METHOD

- Starting from arbitrary values assigned to each node, the computation iterates until convergence is achieved
  - that is, until $|S^{k+1}(N_i) - S^{k}(N_i)| < \mu$
- After running the algorithm, the score associated with each node represents the node's "importance" within the graph

TEXTRANK FOR WEIGHTED GRAPHS

- In case of weighted graphs, where weights represent the strength of the connection between node pairs, weighted node score is:

$$WS(N_i) = (1 - d) + d * \sum_{j \in In(N_i)} \frac{w_{ji}}{\sum_{N_k \in Out(N_j)} w_{kj}} WS(N_j)$$

$WS(N_i)$ – weighted score for node *i*

w_{ij} – weight (strength) of the connection between nodes *i* and *j*

TEXTRANK FOR KEYWORDS EXTRACTION

- The input text is pre-processed
  - tokenization and part-of-speech tagging
- Co-occurrence (undirected) graph is created
  - a node is created for each unique noun and adjective of the input text
  - an edge is added between nodes (i.e. words) that co-occur within a window of N words ($N \in \{2,10\}$)*
- The ranking algorithm is run
  - initial score for all the nodes is set to 1
  - the algorithm is run until the conversion (typically 20-30 iterations) at the chosen threshold (e.g. $\mu = 10^{-4}$)

*The authors' experiments showed that the larger the window, the lower the precision; N=2 proved the best.

TEXTRANK FOR KEYWORDS EXTRACTION (CONT.)

- Nodes are sorted based on their final score, and top *T* (or *T%* of) words are taken as potential keywords
- Post-processing: potential keywords are matched against the input text, and sequences of adjacent keywords are collapsed into multi-word keywords
  - E.g. in the text “Matlab code for plotting functions”, if both *Matlab* and *code* are among the potential keywords, they would be collapsed into *Matlab code*

TEXTRANK FOR TEXT SUMMARIZATION

TextRank method can be also used for extracting relevant sentences from the input text, thus, effectively enabling automated text summarization

In this application case:

- nodes of the graph are whole sentences
- edges are established based on the sentence *similarity*

TextRank for text summarization (cont.)

- The intuition:
  - the similarity relation between two sentences can be seen as a act of “recommendation”: a sentence recommends other sentences that address similar concepts
  - the sentences that are highly recommended by other sentences in the text are likely to be more informative for the given text

TEXTRANK FOR TEXT SUMMARIZATION (CONT.)

- Sentence similarity can be measured in many different ways
  - E.g., cosine similarity, longest common subsequence, various string metrics

- The authors' original proposal is based on the content (word) overlap of two sentences S_i and S_j:

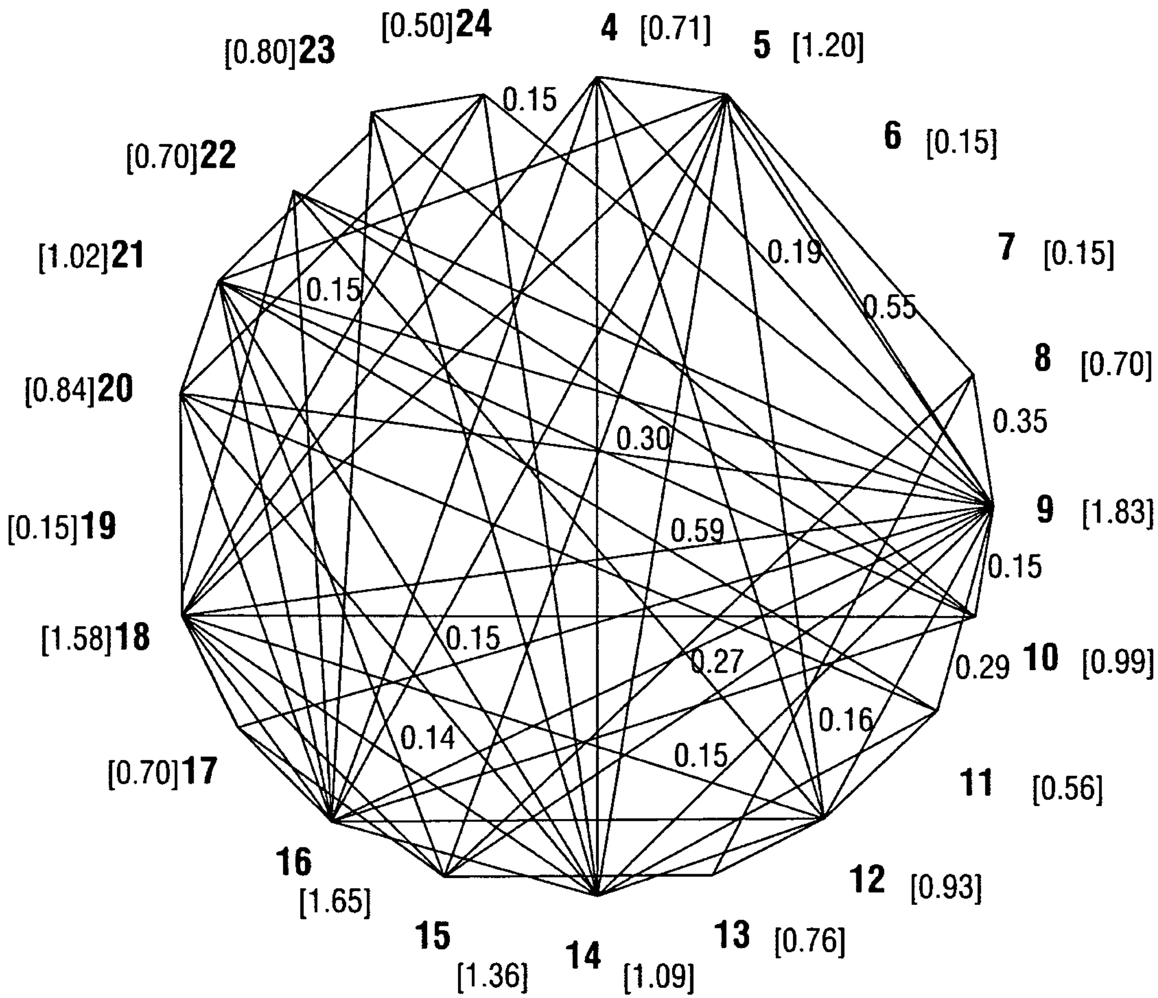
$$Similarity(S_i, S_j) = \frac{|\{w_k | w_k \in S_i \,\& w_k \in S_j\}|}{\log(|S_i|) + \log(|S_j|)}$$

The similarity measure uses the length of the sentences as the normalization factor to avoid promotion of long sentences

TextRank for text summarization (cont.)

- The resulting graph is weighted and highly connected
  - edge weights correspond to the computed similarities of the text sentences
  - graph density can be reduced by setting the minimum similarity value for establishing a connection
- The (weighted) ranking algorithm is run on the graph
- Sentences are sorted based on their score
- The top ranked sentences are selected for the summary

EXAMPLE WEIGHTED SENTENCE GRAPH



Source: https://www.google.com/patents/US7809548

IMPLEMENTATION OF TEXTRANK

- TextRank method is patented:
  https://www.google.com/patents/US7809548

- No 'official' implementation, but several implementations in different programing languages (Java, Python, R,…)
  - Easy to find by googling it