TEXT MINING: FROM NAIVE BAYES TO LDA

Sergey Nikolenko

Harbour Space University, Barcelona, Spain

March 30, 2017

NATURAL LANGUAGE PROCESSING

- A key problem for artificial intelligence, recognized very early.
- Early optimism: 1950s, Noam Chomsky, Dartmouth seminar.
- But proved to be not so easy: the first winter of neural networks was in part due to a fail of a machine translation project.
- And we are still far from understanding text.

- Why is it hard? In part, due to our model of the world, commonsense reasoning.
- Example — *anaphora* resolution:
    - the suitcase did not fit in the trunk because it was too big;
    - the suitcase did not fit in the trunk because it was too small.
- This is a very well defined problem, a classification problem basically, but it's extremely hard to get to human level.
- What are NLP problems in general?

(1) Syntactic problems:

- *part-of-speech tagging*: label parts of speech (noun, verb, adjective…) and morphology (gender, case…);

(1) Syntactic problems:
  - *part-of-speech tagging*;
  - *morphological segmentation*: divide words into *morphemes*, i.e., prefixes, suffixes and the such;

(1) Syntactic problems:

- *part-of-speech tagging*;
- *morphological segmentation*;
- *stemming* and/or *lemmatization*: reducing a word to its basic form;

(1) Syntactic problems:
  - *part-of-speech tagging*;
  - *morphological segmentation*;
  - *stemming*
  - *sentence boundary disambiguation*: “*House, M.D.* is a favourite TV series of George R. R. Martin”; in languages like Chinese even *word segmentation* is hard;

(1) Syntactic problems:
- *part-of-speech tagging*;
- *morphological segmentation*;
- *stemming*
- *word/sentence boundary disambiguation*;
- *named entity recognition*: find proper names in the text and find out what kind of entities they represent;

(1) Syntactic problems:
- *part-of-speech tagging*;
- *morphological segmentation*;
- *stemming*
- *word/sentence boundary disambiguation*;
- *named entity recognition*;
- *word sense disambiguation*, i.e., homonymy; common sense again:
    - she cannot bear children, they are too noisy;
    - she cannot bear children due to an unfortunate operation.

NLP TASKS

(1) Syntactic problems:

- *part-of-speech tagging*;
- *morphological segmentation*;
- *stemming*
- *word/sentence boundary disambiguation*;
- *named entity recognition*;
- *word sense disambiguation*;
- *syntactic parsing*: construct a syntax tree (dependency tree) of a sentence;

(1) Syntactic problems:
- *part-of-speech tagging*;
- *morphological segmentation*;
- *stemming*
- *word/sentence boundary disambiguation*;
- *named entity recognition*;
- *word sense disambiguation*;
- *syntactic parsing*;
- *coreference resolution*: which objects various words refer to; anaphora is a special case.

(1) Syntactic problems.

(2) Well defined semantic problems:

- *language modeling*: predict the next word or symbol; this is very important, e.g., for speech recognition;

(1) Syntactic problems.

(2) Well defined semantic problems:
   - *language modeling*;
   - *sentiment analysis*: find out whether the text speaks positively or negatively about its subject;

(1) Syntactic problems.

(2) Well defined semantic problems:

- *language modeling*;
- *sentiment analysis*;
- *relationship/fact extraction*: extract well-defined relations or facts from the text, e.g., who married whom, which year the company was founded, and so on;

(1) Syntactic problems.

(2) Well defined semantic problems:

- *language modeling*;
- *sentiment analysis*;
- *relationship/fact extraction*;
- *question answering*: either pure classification (multiple choice), classification with very large label space (trivia questions), or even text generation (questions in a dialogue).

(1) Syntactic problems.
(2) Well defined semantic problems.
(3) Not really well defined semantic problems.
    - *text generation*;

(1) Syntactic problems.
(2) Well defined semantic problems.
(3) Not really well defined semantic problems.
    - *text generation*;
    - *automatic summarization*: generate an abstract for a paper; again, can be a classification problem if we just choose the most representative sentences from the text itself;

(1) Syntactic problems.
(2) Well defined semantic problems.
(3) Not really well defined semantic problems.
    - *text generation*;
    - *automatic summarization*;
    - *machine translation*;

(1) Syntactic problems.
(2) Well defined semantic problems.
(3) Not really well defined semantic problems.
    - *text generation*;
    - *automatic summarization*;
    - *machine translation*;
    - *dialog and conversational models*: talk to a human or at least answer her questions.

- The problem often can be frames as *text categorization* (*classification*).
- We can use regular classifiers (logistic regression, SVM); but what are the inputs?
- One approach: bag of words.
- Why could it be less than perfect?

- One reason: it will depend on the words that are not really important.
- Variation – *tf-idf weights*:

$$\mathrm{tf}(t,d) = \frac{n_t}{|d|}, \quad \mathrm{idf}(t,D) = \log \frac{|D|}{|\{d \in D \mid t \in d\}|}.$$

- Usually the result improves if we use tf-idf weights.

NAIVE BAYES

- Given a set of texts divided into categories, train the model to classify new texts.
- Attributes $a_1, a_2, \ldots, a_n$ are words, v is the text topic/label; we will use the bag of words approach for now.

- This is already a huge simplification, but we still are not able to directly estimate $p(a_1, a_2, \dots, a_n | x = v)$.
- We need more simplifying assumptions.
- Naive Bayes classifier – the simplest model: assume all words are conditionally independent given the category:

$$p(a_1, a_2, \dots, a_n | x = v) = p(a_1 | x = v) p(a_2 | x = v) \dots p(a_n | x = v).$$

- And choose v as

$$v_{NB}(a_1, a_2, \dots, a_n) = \arg\max_{v \in V} p(x = v) \prod_{i=1}^{n} p(a_i | x = v).$$

- Despite crazy assumptions, works rather well in practice, there are reasons for this.

- Two variations: multinomial and multivariate naive Bayes.
- In the multivariate model a document is a vector of binary attributes: does a word occur in the text?
- And the likelihood is basically multidimensional Bernoulli trials (coin tosses).
- The “naive” assumption is that these coins are assumed to be independent.

- Let $V = \{w_t\}_{t=1}^{|V|}$ be the dictionary. Then d_i is a vector of length $|V|$ consisting of bits B_{it}; $B_{it} = 1$ iff word w_t occurs in document d_i.
- Likelihood of the event that d_i is from class c_j:

$$p(d_i \mid c_j) = \prod_{t=1}^{|V|} \left(B_{it} p(w_t \mid c_j) + (1 - B_{it})(1 - p(w_t \mid c_j)) \right).$$

- To train this classifier we need to train the probabilities $p(w_t \mid c_j)$.

- Learning is easy: given a set of documents $D = \{d_i\}_{i=1}^{|D|}$ with labels c_j and vocabulary $V = \{w_t\}_{t=1}^{|V|}$, we know the bits B_{it} and can simply derive (with Laplace smoothing – smoothing is important here!):

$$p(w_t \mid c_j) = \frac{1 + \sum_{i=1}^{|D|} B_{it} p(c_j \mid d_i)}{2 + \sum_{i=1}^{|D|} p(c_j \mid d_i)}.$$

- Prior probabilities are also easy: $p(c_j) = \frac{1}{|D|}\sum_{i=1}^{|D|} p(c_j \mid d_i)$.
- And the classification is done as

$$c = \arg\max_j p(c_j) p(d_i \mid c_j) =$$
$$= \arg\max_j \left(\frac{1}{|D|}\sum_{i=1}^{|D|} p(c_j \mid d_i)\right) \prod_{t=1}^{|V|} \left(B_{it} p(w_t \mid c_j) + (1 - B_{it})(1 - p(w_t \mid c_j))\right) =$$
$$= \arg\max_j \left(\log(\sum_{i=1}^{|D|} p(c_j \mid d_i)) + \sum_{t=1}^{|V|} \log\left(B_{it} p(w_t \mid c_j) + (1 - B_{it})(1 - p(w_t \mid c_j))\right)\right)$$

- In the multinomial model a document is a set of words drawn from a bag with replacement, like rolling a really huge die.
- The likelihood now accounts for numbers of occurrences but does not account for the words that are *not* there.
- For a vocabulary $V = \{w_t\}_{t=1}^{|V|}$, the document d_i is a vector of length $|d_i|$ consisting of words taken with probability $p(w_t \mid c_j)$.
- Likelihood of the event that d_i is from class c_j:

$$p(d_i \mid c_j) = p(|d_i|)|d_i|! \prod_{t=1}^{|V|} \frac{1}{N_{it}!} p(w_t \mid c_j)^{N_{it}},$$

  where N_{it} is the number of occurrences of w_t in d_i.
- To train this classifier we need to train the probabilities $p(w_t \mid c_j)$.

- Learning is still easy: given a set of documents $D = \{d_i\}_{i=1}^{|D|}$ with labels c_j and vocabulary $V = \{w_t\}_{t=1}^{|V|}$, we know N_{it} and can compute (again with smoothing)

$$p(w_t \mid c_j) = \frac{1 + \sum_{i=1}^{|D|} N_{it} p(c_j \mid d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{is} p(c_j \mid d_i)}.$$

- Prior probabilities are, again, $p(c_j) = \frac{1}{|D|} \sum_{i=1}^{|D|} p(c_j \mid d_i)$.
- And the classification is done as

$$\begin{aligned} c &= \arg\max{}_j p(c_j) p(d_i \mid c_j) = \\ &= \arg\max{}_j \left(\frac{1}{|D|} \sum_{i=1}^{|D|} p(c_j \mid d_i) \right) p(|d_i|) |d_i|! \prod_{t=1}^{|V|} \frac{1}{N_{it}!} p(w_t \mid c_j)^{N_{it}} = \\ &= \arg\max{}_j \left(\log \left(\sum_{i=1}^{|D|} p(c_j \mid d_i) \right) + \sum_{t=1}^{|V|} N_{it} \log p(w_t \mid c_j) \right). \end{aligned}$$

- Naive Bayes has two features that make life much easier:
    - we know the labels of every document;
    - each document has only one label/topic.
- Let us now remove both of these constraints.
- First, what can we do if we *don't* know the labels?

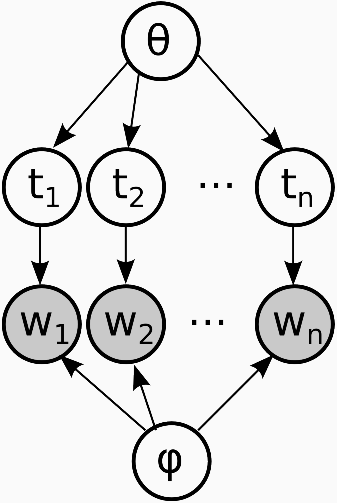
- Then it becomes a clustering problem.
- And we can solve it with the EM algorithm:
    - on the E step we compute expectations of which document belongs to which topic;
    - on the M step we compute, with regular naive Bayes, the probabilities $p(w \mid t)$ for fixed labels.
- This is the easy generalization.
- The interesting one is to account for multiple topics in a single document...

TOPIC MODELING

- Consider the following model:
  - each word in a document d is generated from some topic $t \in T$;
  - a document is generated with some distribution on the topics $p(t \mid d)$;
  - a word is generated from a topic rather than a document: $p(w \mid d, t) = p(w \mid d)$;
  - and as a result we get the following likelihood:

$$p(w \mid d) = \sum_{t \in T} p(w \mid t) p(t \mid d).$$

- This model is called *probabilistic latent semantic analysis*, pLSA (Hoffmann, 1999).


θ
t1
t2
⋯
tn
w1
w2
⋯
wn
φ

- How do we train pLSA? We can estimate $p(w \mid d) = \frac{n_{wd}}{n_d}$, but we actually need
  - $\phi_{wt} = p(w \mid t)$;
  - $\theta_{td} = p(t \mid d)$.
- Maximize the likelihood

$$p(D) = \prod_{d \in D} \prod_{w \in d} p(d, w)^{n_{dw}} = \prod_{d \in D} \prod_{w \in d} \left[\sum_{t \in T} p(w \mid t) p(t \mid d) \right]^{n_{dw}}.$$

- How can we maximize a function like this?

- The EM algorithm! On the E step we find how many words w were generated in document d by topic t:

$$n_{dwt} = n_{dw} p(t \mid d, w) = n_{dw} \frac{\phi_{wt}\theta_{td}}{\sum_{s \in T} \phi_{ws}\theta_{sd}}.$$

- On the M-step, recompute model parameters:

$$\begin{array}{lll} n_{wt} = \sum_d n_{dwt}, & n_t = \sum_w n_{wt}, & \phi_{wt} = \frac{n_{wt}}{n_t}, \\ n_{td} = \sum_{w \in d} n_{dwt}, & \theta_{td} = \frac{n_{td}}{n_d}. & \end{array}$$

- And that's it about inference in pLSA.

- We don't even have to store the whole matrix n_{dwt}; we can iterate over the documents, adding n_{dwt} to the global counters n_{wt}, n_{td}.
- What's missing?
  - First, there are exponentially many local optima.
  - Second, there are really many parameters; we're definitely heading for overfitting;
  - Third, it would be great to find not just some optimum but an optimum with good, desirable properties.
- How can we achieve all this?

- With regularization! Lots of different regularizers for pLSA (ARTM).
- In general we just add R_i to the log likelihood:

$$\sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_{t \in T} \phi_{wt} \theta_{td} + \sum_i \tau_i R_i(\Phi, \Theta).$$

- Then in the EM algorithm we will have partial derivatives of R on the M-step:

$$n_{wt} = \left[\sum_{d \in D} n_{dwt} + \phi_{wt} \frac{\partial R}{\partial \phi_{wt}} \right]_+ ,$$

$$n_{td} = \left[\sum_{w \in d} n_{dwt} + \theta_{td} \frac{\partial R}{\partial \theta_{td}} \right]_+$$

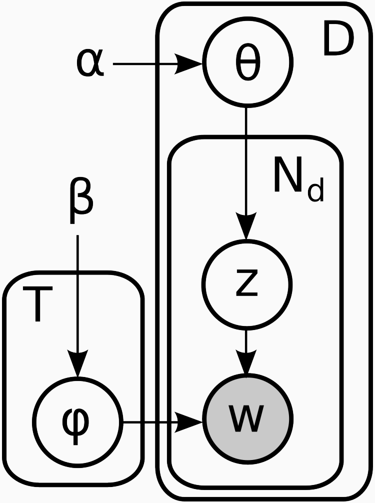
- To prove this, we consider EM as solving an optimization problem with Karush–Kuhn–Tucker conditions.

- And we can now mix and match lots of different regularizers:
  - smoothing regularizer (later; similar to LDA);
  - sparsity regularizer: maximize KL-divergence between distributions ϕ_{wt} and θ_{td} and the uniform distribution;
  - constrasting regularizer: minimize the covariances between vectors ϕ_t, so that each topic would get its own “lexical kernel”, i.e., characteristic words;
  - coherence regularizer: give bonuses for words that are close to each other in documents;
  - and so on, we can have many more ideas.

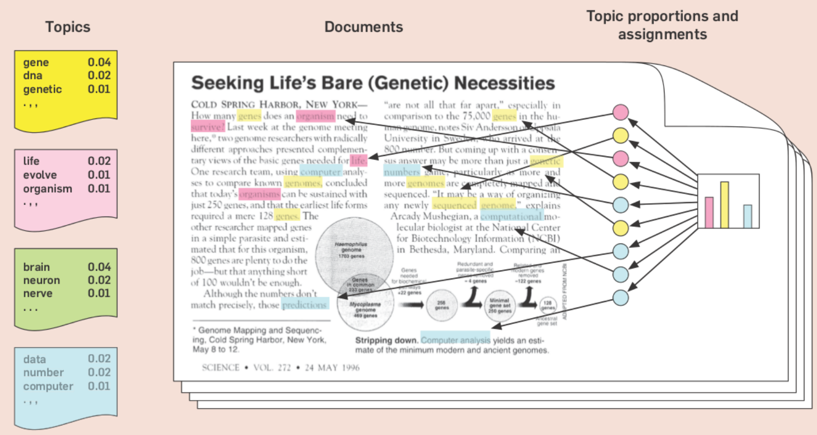
- Extension of pLSA ideas – LDA (Latent Dirichlet Allocation).
- Bayesian version of pLSA; we add the priors and perform approximate Bayesian inference.
- The problem is the same: model a large corpus of texts.

- One document may have several topics. We construct a hierarchical model:
  - on the first level – a mixture whose components correspond to "topics";
  - on the second level – a multinomial variable with Dirichlet prior that defines the "distribution of topics" in a document.

- Formally: words are taken from a dictionary $\{1, \dots, V\}$; a word is a vector w, $w_i \in \{0, 1\}$, where exactly one component equals 1.
- A document is a sequence of N words $\mathbf{w}$. We are given a corpus of M documents $\mathcal{D} = \{\mathbf{w}_d \mid d = 1..M\}$.
- The LDA generative model:
  - sample $\theta \sim \mathrm{Di}(\alpha)$;
  - for each of N words w_n:
    - sample topic $z_n \sim \mathrm{Mult}(\theta)$;
    - sample word $w_n \sim p(w_n \mid z_n, \beta)$ по мультиномиальному распределению.


α
θ
D
N_d
β
T
z
φ
w

LDA: THE RESULTS [BLEI, 2012]



- Two main approaches to inference in complex probabilistic models, including LDA:
    - *variational approximations*: consider a family of simpler distributions with new parameters and find the best approximation (will discuss later);
    - *sampling*: generate points from a complex distribution without computing it explicitly, by running a Markov chain under the density graph (Gibbs sampling is a special case).
- Gibbs sampling is usually easier to extend, but a good variational approximation works faster and can be more stable.

LDA: GIBBS SAMPLING

- In the basic LDA model, Gibbs sampling after simple transformations reduces to the so-called *collapsed Gibbs sampling*, where variables z_w are iteratively sampled as

$$p(z_w = t \mid \mathbf{z}_{-w}, \mathbf{w}, \alpha, \beta) \propto q(z_w, t, \mathbf{z}_{-w}, \mathbf{w}, \alpha, \beta) =$$

$$\frac{n^{(d)}_{-w,t} + \alpha}{\sum_{t' \in T} \left(n^{(d)}_{-w,t'} + \alpha \right)} \frac{n^{(w)}_{-w,t} + \beta}{\sum_{w' \in W} \left(n^{(w')}_{-w,t} + \beta \right)},$$

  where $n^{(d)}_{-w,t}$ is the number of words in document d generated by topic t, and $n^{(w)}_{-w,t}$ is the number of times w was generated from topic t except the current value z_w; note that both these counters depend on the other variables $\mathbf{z}_{-w}$.

- With these samples we can then estimate model parameters:

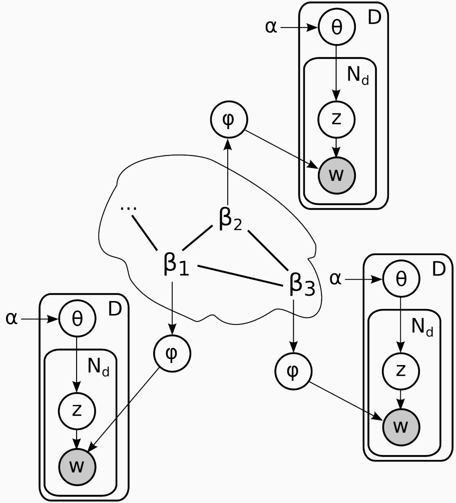
$$\theta_{d,t} = \frac{n^{(d)}_{-w,t} + \alpha}{\sum_{t' \in T} \left(n^{(d)}_{-w,t'} + \alpha \right)},$$

$$\phi_{w,t} = \frac{n^{(w)}_{-w,t} + \beta}{\sum_{w' \in W} \left(n^{(w')}_{-w,t} + \beta \right)},$$

  where $\phi_{w,t}$ is the probability to get word w in topic t, and $\theta_{d,t}$ is the probability to get topic t in document d.
- Next – LDA extensions...

- In the basic LDA model, word-topic distributions are independent and uncorrelated; this is not true in practice, of course.
- Correlated topic models (correlated topic models, CTM); we use logistic normal distribution instead of the Dirichlet prior, and we can now model correlations between topics.
- Markov topic models (MTM): Markov random fields (undirected models) to model the relations between topics in different parts of the dataset (e.g., different corpora).
- MTM has several copies of hyperparameters β_i related in a Markov random field (MRF). Texts from the ith corpus are generated as in regular LDA with the corresponding β_i.
- In turn, β_i are subject to prior constraints that “divide” the topics between corpora, specify “background” topics and so on.

MARKOV TOPIC MODELS



- Relational topic model (RTM): a hierarchical model that reflects the structural graph of a network of documents.
- Generative process in RTM:
  - generate the documents from a regular LDA model;
  - for every pair of documents d_1, d_2 choose a binary variable y_{12} that reflects the relation between d_1 and d_2:

$$y_{12} \mid \mathbf{z}_{d_1}, \mathbf{z}_{d_2} \sim \psi(\cdot \mid \mathbf{z}_{d_1}, \mathbf{z}_{d_2}, \eta).$$

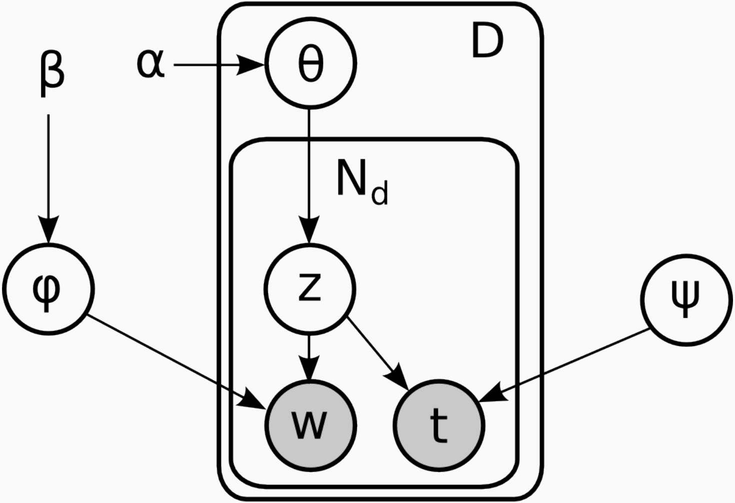
- As ψ one can take various sigmoid functions.

- A number of important extensions aim to account for the trends, i.e., changes in the distributions of topics with time.
- What are the “hot” topics? How do they evolve? Which topics are stable?..

- In the Topics over Time (TOT) model, time is continuous, and the model is augmented with a Beta distribution that generates timestamps for every document.
- The Topics over Time generative model:
  - for every topic $z = 1..T$ sample the multinomial distribution ϕ_z from the Dirichlet prior β;
  - for every document d sample the multinomial distribution θ_d from the Dirichlet prior α;
  - then for each word $w_{di} \in d$:
    - sample topic z_{di} из θ_d;
    - sample word w_{di} from distribution $\phi_{z_{di}}$;
    - sample time t_{di} from the beta distribution $\psi_{z_{di}}$.

- Each topic corresponds to its own beta distribution ψ_z, i.e., topics are localized in time (depending on parameters ψ_z).
- Thus, we can both train global topics that are always present and find a topic that has had a short burst and then disappeared; for the latter the variance of ψ_z will be smaller than for the former.

TOPICS OVER TIME

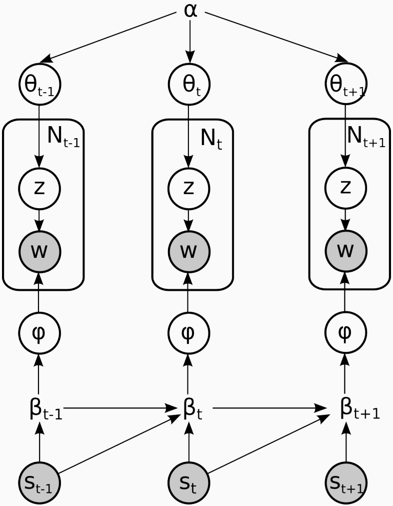


- *Dynamic topic models* represent temporal evolution through changing hyperparameters α and/or β.
- Discrete ([d]DTM), where time is discrete, and continuous, where the evolution of hyperparameter β (α is assumed constant) is modeled with Brownian motion: for two documents i and j (j is later than i)

$$\beta_{j,k,w} \mid \beta_{i,k,w}, s_i, s_j \sim \mathcal{N}(\beta_{i,k,w}, v\Delta_{s_i,s_j}),$$

  where s_i and s_j are timestamps of documents i and j, $\Delta(s_i, s_j)$ is the time interval between them, v is the model parameter.
- Otherwise the generative process is the same.

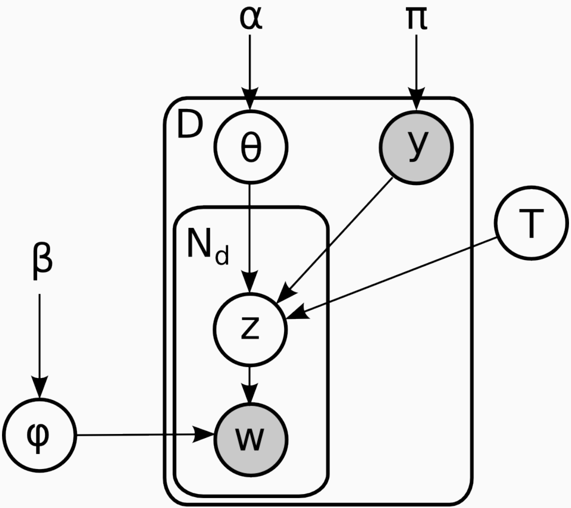
CONTINUOUS DTM



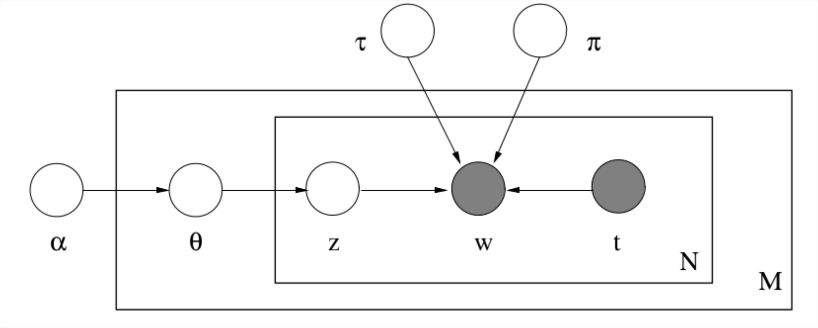
- Supervised LDA: documents have additional information, a response variable.
- The response distribution is modeled with a generalized linear model whose parameters are related with the document-topic distribution.
- I.e., in the generative model, after topics are known for a document, we
  - generate the response variable $y \sim \mathrm{glm}(\mathbf{z}, \eta, \delta)$, where $\mathbf{z}$ is the distribution of topics in the document, and η and δ are other glm parameters.
- E.g., in recommender systems it could be the user's reaction.

DiscLDA

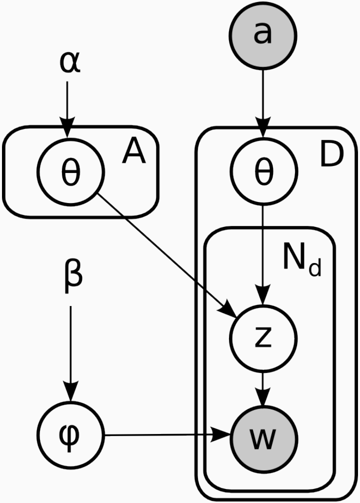
- Discriminative LDA (DiscLDA): another extension of LDA for documents with a categorial variable y which will become a classification target.
- For every class label y DiscLDA introduces a linear transformation $T^y : \mathbb{R}^K \to \mathbb{R}_+^L$, that maps K-dimensional Dirichlet prior θ to a mixture of L-dimensional Dirichlet distributions $T^y\theta$.
- Only the step of generating the topics z for a document changes: instead of choosing z by distribution θ generated for this document, we generate topic z by distribution $T^y\theta$, where T^y is a transformation corresponding to the label y for the current document.


α
π
D
θ
y
T
N_d
β
z
φ
w

- TagLDA: words have tags, i.e., a document is multiple bags of words with different words in different bags.
- E.g., a web page might have a title, and words from the title are more important. Or actual tags.
- Mathematically, topic-word distributions are not discrete multinomial distributions but factorized into word-topic and word-tag distributions.


τ
π
α
θ
z
w
t
N
M

- Author-Topic modeling: apart from the texts themselves, we have their authors; each author is a distribution on the topics.
- The basic generative Author-Topic model:
    - for each word w:
        - sample author x for this word from the set of authors of document a_d;
        - sample topic from the distribution on the topics corresponding to author x;
        - sample word from the distribution слов corresponding to this topic.


a
α
θ
A
θ
D
N d
β
z
φ
w

- To sample from the AT model, we use a variation of Gibbs sampling:

$$p(z_w = t, x_w = a \mid \mathbf{z}_{-w}, \mathbf{x}_{-w}, \mathbf{w}, \alpha, \beta) \propto$$

$$\propto \frac{n^{(a)}_{-a,t} + \alpha}{\sum_{t' \in T} \left(n^{(a)}_{-w,t'} + \alpha \right)} \frac{n^{(w)}_{-w,t} + \beta}{\sum_{w' \in W} \left(n^{(w')}_{-w,t} + \beta \right)},$$

where $n^{(a)}_{-a,t}$ is how many times author a corresponded to topic t except the current value x_w, $n^{(w)}_{-w,t}$ is how many times word w was generated from topic t except the current value z_w; note that both these counters depend on the other variables $\mathbf{z}_{-w}$, $\mathbf{x}_{-w}$.

1. NLP tasks
2. Classical categorization: naive Bayes classifier.
3. Generalizing naive Bayes: clustering EM-алгоритмом.
4. Topic modeling: pLSA, LDA, LDA extensions.

THANK YOU!

Thank you for your attention!