Università degli Studi di Firenze

Facoltà di Ingegneria

Probabilistic topic models: Latent Dirichlet Allocation

Marco Righini

21 Febbraio 2013

Probabilistic topic models

Collective knowledge (scientific articles, books, images. . .) continues to be digitized and stored.

Growing interest in tools for understanding, organizing and searching information in these vast domains.

Topic models

Statistical models that analyze the words of documents to discover the **hidden thematic structure** and annotate the documents according to that, through an unsupervised approach.

Probabilistic topic models

Collective knowledge (scientific articles, books, images. . .) continues to be digitized and stored.

Growing interest in tools for understanding, organizing and searching information in these vast domains.

Topic models

Statistical models that analyze the words of documents to discover the **hidden thematic structure** and annotate the documents according to that, through an unsupervised approach.

Precursors of LDA

TF-IDF [Salton et Gills, 1983] : identifies sets of words discriminative for a set of documents but doesn't reveal too much of inter or intra documents statistical relations;

LSI [Deerwester et al., 1990] : based on SVD of TF-IDF matrix, derived features can capture some basic linguistic notions such as synonymy and polysemy;

pLSI [Hofmann, 1999] : documents are reduced to a probability distribution on a fixed set of topics. Still there's no generative probabilistic model at level of documents.

Basic intuition

Assumptions of dimensionality reduction methods: exchangeability of words in a document and of documents in the corpus.

Theorem (De Finetti). *Any collection of infinitely exchangeable random variables has a representation as mixture distribution.*

Idea [Blei, Ng and Jordan, 2003]

Represent documents as random mixture over latent topics and each topic as a distribution over terms.

Basic intuition

Assumptions of dimensionality reduction methods: exchangeability of words in a document and of documents in the corpus.

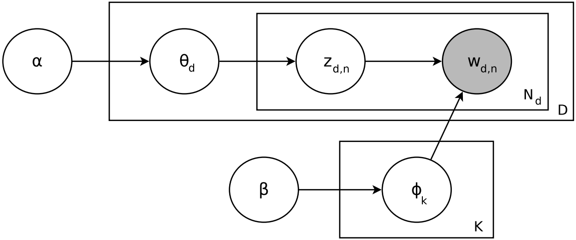
Theorem (De Finetti). *Any collection of infinitely exchangeable random variables has a representation as mixture distribution.*

Idea [Blei, Ng and Jordan, 2003]

Represent documents as random mixture over latent topics and each topic as a distribution over terms.

Generative process

for $k = 1 \to K$ **do**
    sample $\phi_k \sim Dirichlet(\beta)$
end for
for $d = 1 \to D$ **do**
    sample $\theta_d \sim Dirichlet(\alpha)$
    for $n = 1 \to N_d$ **do**
        sample $z_{d,n} \sim Multinomial(\theta_d)$
        sample $w_{d,n} \sim Multinomial(\phi_{z_{d,n}})$
    end for
end for



Dirichlet distribution

A Dirichlet distribution $\theta \sim Dirichlet(\alpha_1, \ldots, \alpha_K)$ is a distribution over the K-1 simplex, i.e. vectors of K components that sum to one.

It has some important properties:

- **conjugate prior** of categorical and multinomial distributions;
- **mean** vector m with $m_i = \mathbb{E}[\theta_i | \alpha_1, \ldots, \alpha_K] = \frac{\alpha_i}{\sum_{k=1}^K \alpha_k}$;
- **scale** $s = \sum_{k=1}^K \alpha_k$ controls the peakedness around the mean.

Typically **symmetric Dirichlet distribution** ($\alpha_i = \alpha$ for $i = 1 \rightarrow K$) is chosen when there is no prior knowledge favoring one component over another.

$\alpha = 1$ defines an uniform distribution, $\alpha << 1$ and $\alpha >> 1$ respectively a sparser or more dense distribution.

Dirichlet distribution

A Dirichlet distribution $\theta \sim \mathit{Dirichlet}(\alpha_1, \ldots, \alpha_K)$ is a distribution over the K-1 simplex, i.e. vectors of K components that sum to one.

It has some important properties:

- **conjugate prior** of categorical and multinomial distributions;
- **mean** vector m with $m_i = \mathbb{E}[\theta_i|\alpha_1, \ldots, \alpha_K] = \frac{\alpha_i}{\sum_{k=1}^{K} \alpha_k}$;
- **scale** $s = \sum_{k=1}^{K} \alpha_k$ controls the peakedness around the mean.

Typically **symmetric Dirichlet distribution** ($\alpha_i = \alpha$ for $i = 1 \to K$) is chosen when there is no prior knowledge favoring one component over another.

$\alpha = 1$ defines an uniform distribution, $\alpha << 1$ and $\alpha >> 1$ respectively a sparser or more dense distribution.

Approximate posterior inference

Joint distribution:

$$p(\mathbf{w}, \mathbf{z}, \theta, \phi|\alpha, \beta) = \prod_{k=1}^{K} p(\phi_k|\beta) \prod_{d=1}^{D} p(\theta_d|\alpha)(\prod_{n=1}^{N_d} p(z_{d,n}|\theta_d)p(w_{d,n}|\phi_{z_{d,n}}))$$

Posterior distribution $p(\mathbf{z}, \theta, \phi|\mathbf{w}, \alpha, \beta) = \dfrac{p(\mathbf{w}, \mathbf{z}, \theta, \phi|\alpha, \beta)}{p(\mathbf{w}|\alpha, \beta)}$

Theorically, the evidence can be obtained marginalizing the joint distribution over all possible configurations of the hidden variables. In practice, exact inference is intractable because the configurations domain is too large.

Some algorithms have been developed to efficiently **approximate** the posterior:

- *sampling-based algorithms* (Gibbs sampling);
- *variational algorithms*.

Approximate posterior inference

Joint distribution:

$$p(\mathbf{w}, \mathbf{z}, \theta, \phi|\alpha, \beta) = \prod_{k=1}^{K} p(\phi_k|\beta) \prod_{d=1}^{D} p(\theta_d|\alpha)(\prod_{n=1}^{N_d} p(z_{d,n}|\theta_d)p(w_{d,n}|\phi_{z_{d,n}}))$$

Posterior distribution $p(\mathbf{z}, \theta, \phi|\mathbf{w}, \alpha, \beta) = \dfrac{p(\mathbf{w}, \mathbf{z}, \theta, \phi|\alpha, \beta)}{p(\mathbf{w}|\alpha, \beta)}$

Theorically, the evidence can be obtained marginalizing the joint distribution over all possible configurations of the hidden variables. In practice, exact inference is intractable because the configurations domain is too large.

Some algorithms have been developed to efficiently **approximate** the posterior:

- *sampling-based algorithms* (Gibbs sampling);
- *variational algorithms*.

Approximate posterior inference

Joint distribution:

$$p(\mathbf{w}, \mathbf{z}, \theta, \phi | \alpha, \beta) = \prod_{k=1}^{K} p(\phi_k | \beta) \prod_{d=1}^{D} p(\theta_d | \alpha) (\prod_{n=1}^{N_d} p(z_{d,n} | \theta_d) p(w_{d,n} | \phi_{z_{d,n}}))$$

Posterior distribution $p(\mathbf{z}, \theta, \phi | \mathbf{w}, \alpha, \beta) = \dfrac{p(\mathbf{w}, \mathbf{z}, \theta, \phi | \alpha, \beta)}{p(\mathbf{w} | \alpha, \beta)}$

Theorically, the evidence can be obtained marginalizing the joint distribution over all possible configurations of the hidden variables. In practice, exact inference is intractable because the configurations domain is too large.

Some algorithms have been developed to efficiently **approximate** the posterior:

- *sampling-based algorithms* (Gibbs sampling);
- *variational algorithms*.

Gibbs sampling

Gibbs sampling is an MCMC (Markov Chain Monte Carlo) algorithm based on the definition of a Markov Chain whose stationary distribution is the posterior of interest.

1: choose (e.g. at random) the initial values $x_1^{(0)}, x_2^{(0)}, \ldots, x_p^{(0)}$ for the p latent variables
2: **for** $t = 1 \to$ T (number of scans) **do**
3: **for** $i = 1 \to p$ **do**
4: sample $x_i^{(t)} \sim p(x_i | x_1^{(t)}, \ldots, x_{i-1}^{(t)}, x_{i+1}^{(t-1)}, \ldots, x_p^{(t-1)})$
5: **end for**
6: **end for**

Once the chain has "burned in", the posterior can be approximated collecting samples at a certain lag (to reduce correlation) and averaging them.

Gibbs sampling

Gibbs sampling is an MCMC (Markov Chain Monte Carlo) algorithm based on the definition of a Markov Chain whose stationary distribution is the posterior of interest.

```
choose (e.g. at random) the initial values x_1^(0), x_2^(0), ..., x_p^(0) for the p latent variables
for t = 1 → T (number of scans) do
    for i = 1 → p do
        sample x_i^(t) ~ p(x_i | x_1^(t), ..., x_{i-1}^(t), x_{i+1}^(t-1), ..., x_p^(t-1))
    end for
end for
```

Once the chain has "burned in", the posterior can be approximated collecting samples at a certain lag (to reduce correlation) and averaging them.

Implementation overview

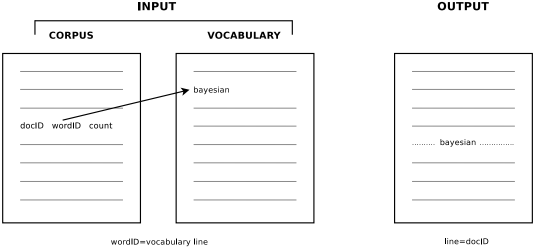
Development has embraced the following phases:

- project of the preprocessing module, including dataset format conversion and k-fold creation;
- project of the LDA module which cares about collapsed Gibbs sampling execution and perplexity computation;
- implementation.

Preprocessing: dataset format conversion

The preprocessing module carries out these tasks:

- **dataset format conversion**



Input dataset was obtained through tokenization, after removal of stop words and rare words.

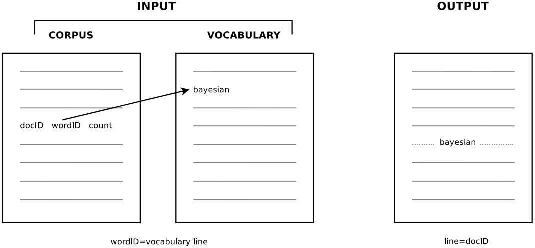
Benefits from textual format

- not all terms in the vocabulary were present in the corpus;
- immediate view of documents content.

Preprocessing: dataset format conversion

The preprocessing module carries out these tasks:

- **dataset format conversion**



Input dataset was obtained through tokenization, after removal of stop words and rare words.

Benefits from textual format

- not all terms in the vocabulary were present in the corpus;
- immediate view of documents content.

Preprocessing: k-folds and usage

- **k-folds creation**: the documents are randomically shuffled to prevent possible sequential correlations in the original dataset. Afterwards, k train-validation pairs are generated.

```
Usage:
createKFold [-h] [--test_frac TEST_FRAC] [--k K] [--output OUTPUT] [--debug] corpus vocabulary

positional arguments:
  corpus                path to the corpus file
  vocabulary            path to the vocabulary file

optional arguments:
  -h, --help            show this help message and exit
  --test_frac TEST_FRAC  fraction of corpus documents to retain as test set (default: 0)
  --k K                 number of folds (default: 1, no validation set)
  --output OUTPUT       output files directory (default: same directory of input)
  --debug               debug flag (default: false)
```

Preprocessing: k-folds and usage

- **k-folds creation**: the documents are randomically shuffled to prevent possible sequential correlations in the original dataset. Afterwards, k train-validation pairs are generated.

```
Usage:
createKFold [-h] [--test_frac TEST_FRAC] [--k K] [--output OUTPUT] [--debug] corpus vocabulary

positional arguments:
  corpus                  path to the corpus file
  vocabulary              path to the vocabulary file

optional arguments:
  -h, --help              show this help message and exit
  --test_frac TEST_FRAC   fraction of corpus documents to retain as test set (default: 0)
  --k K                   number of folds (default: 1, no validation set)
  --output OUTPUT         output files directory (default: same directory of input)
  --debug                 debug flag (default: false)
```

LDA module: collapsed Gibbs sampling

The LDA module performs two tasks:

- **collapsed Gibbs sampling**: obtained from standard Gibbs sampling marginalizing over θ and ϕ;

Data: words $\mathbf{w} \in$ documents $\mathbf{d}$
Result: topic assignments $\mathbf{z}$ and counters $\mathbf{n_{d,k}}$, $\mathbf{n_{k,w}}$ and $\mathbf{n_k}$

begin
    randomly initialize $\mathbf{z}$ and increment counters;
    foreach *iteration* **do**
        foreach $w \in \mathbf{w}$ **do**
            $topic \leftarrow \mathbf{z}[w]$;
            $n_{d,topic}$-=1; $n_{topic,w}$-=1; n_{topic}-=1;
            for $k = 0 \rightarrow K\text{-}1$ **do**
                $p(z = k|\cdot) = (n_{d,k} + \alpha_k)\frac{n_{k,w}+\beta_w}{n_k+\beta \times V}$;
            end
            $topic \leftarrow$ sample from $p(z|\cdot)$;
            $\mathbf{z}[w] \leftarrow topic$;
            $n_{d,topic}$+=1; $n_{topic,w}$+=1; n_{topic}+=1;
        end
    end
    return $\mathbf{z}$, $\mathbf{n_{d,k}}$, $\mathbf{n_{k,w}}$ *and* $\mathbf{n_k}$
end

$$\theta_{d,k} = \frac{n_{d,k}+\alpha_k}{\sum_{z=0}^{K-1}(n_{d,z}+\alpha_z)}$$

$$\phi_{k,w} = \frac{n_{k,w}}{\sum_{v=1}^{|V|}(n_{k,v}+\beta_v)}$$

LDA module: perplexity computation

- **perplexity computation**: perplexity is an indicator of the generalization performance of a model;

Given a new set of documents D_{new}

$$perpl(D_{new}) = exp\left(-\frac{log(p(\mathbf{w_{new}}|\phi))}{n_{new}}\right)$$

where

$$\log(p(\mathbf{w_{new}}|\phi)) = \sum_{d=1}^{|D_{new}|} \sum_{w \in D_{new}^{(d)}} log(p(w|\phi)) =$$

$$= \sum_{d=1}^{|D_{new}|} \sum_{w \in D_{new}^{(d)}} log\left(\sum_{z} p(w|z,\phi)p(z)\right) = \sum_{d=1}^{|D_{new}|} \sum_{w \in D_{new}^{(d)}} log\left(\sum_{k=0}^{K-1} \phi_{k,w}\theta_{d,k}\right)$$

θ for the new documents is obtained running collapsed Gibbs sampling on a new Markov Chain where the previously inferred counters $\mathbf{n_{k,w}}$ and $\mathbf{n_k}$ (i.e. ϕ) are considered fixed.

LDA module: perplexity computation

- **perplexity computation**: perplexity is an indicator of the generalization performance of a model;

Given a new set of documents D_{new}

$$perpl(D_{new}) = exp\left(-\frac{log(p(\mathbf{w_{new}}|\phi))}{n_{new}}\right)$$

where

$$\log(p(\mathbf{w_{new}}|\phi)) = \sum_{d=1}^{|D_{new}|} \sum_{w \in D_{new}^{(d)}} log(p(w|\phi)) =$$

$$= \sum_{d=1}^{|D_{new}|} \sum_{w \in D_{new}^{(d)}} log\left(\sum_{z} p(w|z,\phi)p(z)\right) = \sum_{d=1}^{|D_{new}|} \sum_{w \in D_{new}^{(d)}} log\left(\sum_{k=0}^{K-1} \phi_{k,w}\theta_{d,k}\right)$$

θ for the new documents is obtained running collapsed Gibbs sampling on a new Markov Chain where the previously inferred counters $\mathbf{n_{k,w}}$ and $\mathbf{n_k}$ (i.e. ϕ) are considered fixed.

LDA module: usage

```
Usage:
lda [-h] [--validation VALIDATION] [--output OUTPUT] [--topics TOPICS] [--alpha ALPHA] [--beta BETA]
[--iters ITERS] [--burnin BURNIN] [--savestep SAVESTEP] [--topwords TOPWORDS] [--perplexity]
[--export_complete] --train TRAIN

Option arguments:
  -h [ --help ]           print usage messages

  -t [ --train ] arg      path to training set file

  -v [ --validation ] arg path to validation set file (default: none). Needed if perplexity flag setted
  -o [ --output ] arg     output files directory (default: same directory of input)
  --topics arg            number of topics (default: 100)
  --alpha arg             alpha hyperparameter value (default: 0.05*averageDocumentLength/#topics)
  --beta arg              beta hyperparameter value (default: 200/#wordsInVocabulary)
  --iters arg             number of Gibbs sampling iterations (default: 1000)
  --burnin arg            number of Gibbs sampling iterations considered burnin (default: iters/2)
  --savestep arg          step at which LDA is saved to harddisk (default: (iters-burnin)/10)
  --topwords arg          number of top words to show for each topic (default: 20)
  --perplexity            exports only perplexity (default: false, export variables). Needs validation
  --export_complete       exports all variables (default: false, export only topwords)
```

Execution time

In the beginning, LDA module was implemented in Python language.

At execution time, it turned out to be too much time consuming also vectorializing the operations (over 4 minutes per iteration in a medium level configuration).

Therefore, implementation moved to C++ language.

Timing: C++ vs Python

C++ implementation is approximately eleven time faster than Python one.

Execution time

In the beginning, LDA module was implemented in Python language.

At execution time, it turned out to be too much time consuming also vectorializing the operations (over 4 minutes per iteration in a medium level configuration).

Therefore, implementation moved to C++ language.

Timing: C++ vs Python

C++ implementation is approximately eleven time faster than Python one.

Tests overview

Three datasets were selected for tests:

- **KOS**: set of 3430 blog entries from `dailykos.com`, an american political blog. Total number of words is 467714;
- **NIPS**: set of 1500 papers from NIPS conferences, for a sum of 1932365 words;
- **ENRON**: set of 39861 emails from Enron Corp, for an aggregate of 6412172 words.

For every dataset, tests concerned the following steps:

1. **models evaluation**: models, corresponding to different values of topic number, were defined and then 4-fold cross-validated;
2. **test on the whole dataset**: model with best performance was qualitatively selected and next applied to the whole dataset.

Tests overview

Three datasets were selected for tests:

- **KOS**: set of 3430 blog entries from `dailykos.com`, an american political blog. Total number of words is 467714;
- **NIPS**: set of 1500 papers from NIPS conferences, for a sum of 1932365 words;
- **ENRON**: set of 39861 emails from Enron Corp, for an aggregate of 6412172 words.

For every dataset, tests concerned the following steps:

1. **models evaluation**: models, corresponding to different values of topic number, were defined and then 4-fold cross-validated;
2. **test on the whole dataset**: model with best performance was qualitatively selected and next applied to the whole dataset.

Tests: cross-validation

During the k-th step of 4-fold cross-validation, collapsed Gibbs sampling is executed on the k-th training set of documents.

At regular intervals, perplexity is computed on the k-th validation set.

Convergence assessment

Convergence of collapsed Gibbs sampling is assessed through the perplexity on the validation set.

Tests: cross-validation

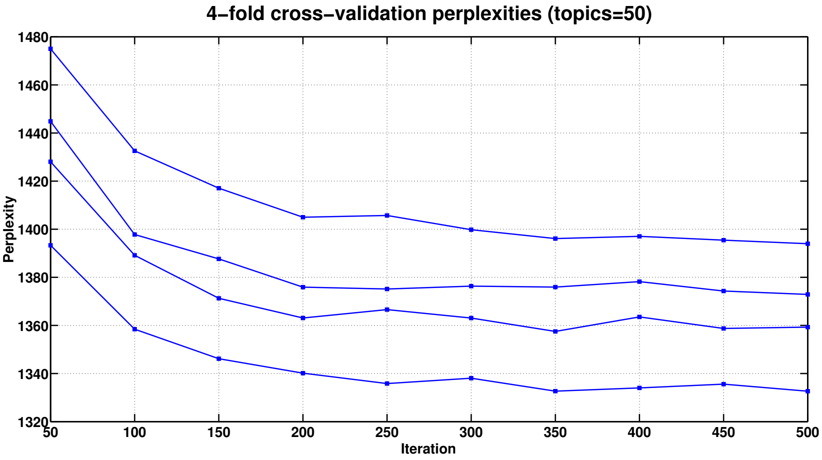
During the k-th step of 4-fold cross-validation, collapsed Gibbs sampling is executed on the k-th training set of documents.

At regular intervals, perplexity is computed on the k-th validation set.

Convergence assessment

Convergence of collapsed Gibbs sampling is assessed through the perplexity on the validation set.

KOS: perplexity

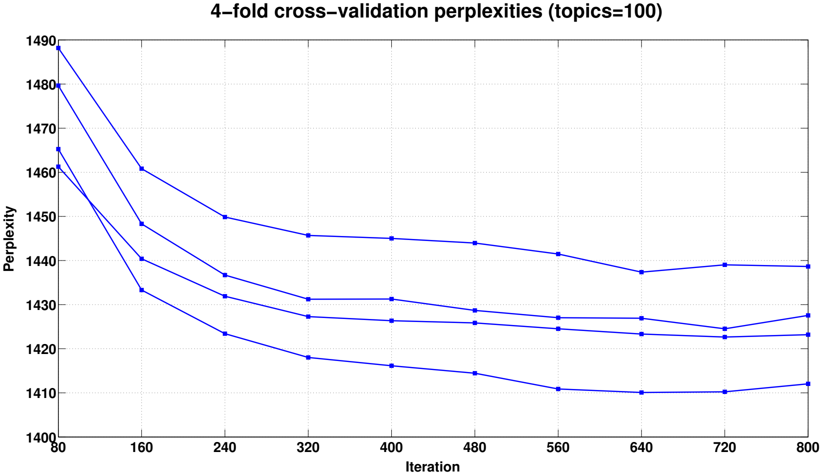


KOS: topics

million	kerry	november	military	iraq
money	john	voting	abu	war
democratic	edwards	election	rumsfeld	weapons
campaign	democratic	house	ghraib	bush
party	general	governor	war	saddam
dnc	presidential	account	iraq	united
raised	kerrys	electoral	prisoners	mass
candidates	party	republicans	prison	threat
fundraising	campaign	senate	soldiers	hussein
national	choice	poll	abuse	powell

cheney	marriage	investigation	jobs	media
bush	rights	law	economy	news
president	gay	federal	economic	times
vice	amendment	justice	growth	press
dick	issue	documents	job	read
administration	law	attorney	rate	article
general	abortion	criminal	numbers	washington
lies	ban	evidence	year	papers
cheneys	civil	allegations	report	local
truth	constitution	source	million	piece

NIPS: perplexity

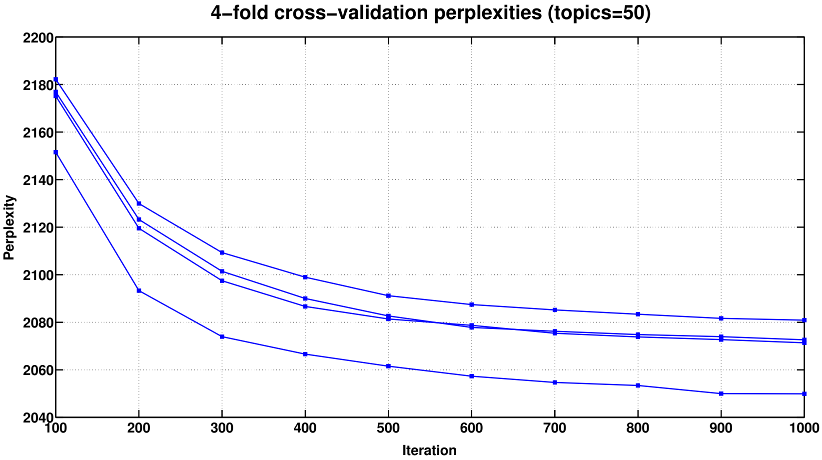


NIPS: topics

network	kernel	prior	gradient	cluster
neural	vector	bayesian	learning	clustering
architecture	support	gaussian	weight	set
output	svm	posterior	descent	rule
feedforward	function	distribution	convergence	group
feed	set	mean	rate	similarity
paper	margin	evidence	stochastic	algorithm
architectures	problem	likelihood	algorithm	partition
forward	examples	covariance	adaptive	structure
type	machines	mackay	perturbation	number

word	face	processor	signal	coding
recognition	human	parallel	frequency	representation
hmm	subject	block	channel	code
speech	recognition	computer	filter	bit
system	representation	memory	power	encoding
training	faces	program	low	vector
context	detection	bit	processing	codes
mlp	facial	performance	spectrum	information
continuous	analysis	machine	detection	decoding
hidden	similarity	operation	frequencies	population

ENRON: perplexity



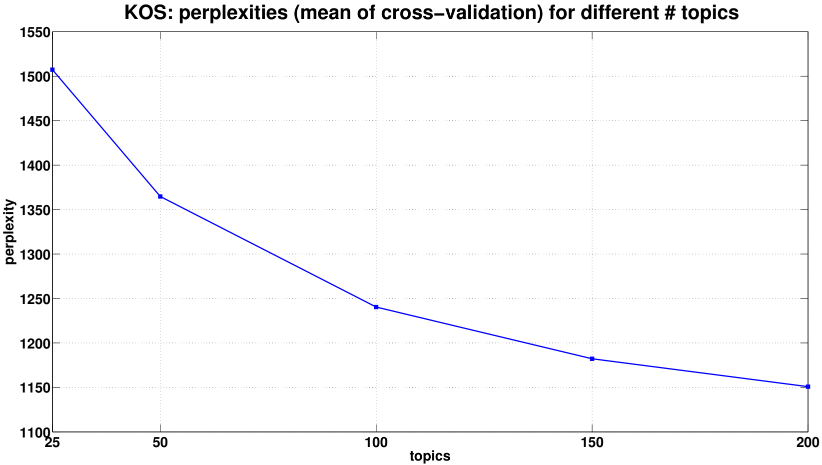
ENRON: topics

energy	project	company	customer	page
program	permit	stock	cost	court
plant	station	financial	rate	labor
air	facility	billion	contract	employees
emission	construction	dynegy	credit	law
environmental	site	investor	rates	worker
demand	water	shares	pay	union
wind	unit	earning	amount	federal
generation	area	analyst	sce	employer
renewable	facilities	trading	period	rules

access	student	american	travel	free
user	business	world	hotel	click
password	school	attack	roundtrip	online
center	program	country	fares	offer
account	university	government	special	receive
web	haas	war	miles	link
link	class	article	city	special
help	mba	bush	ticket	web
message	event	city	offer	visit
site	berkeley	international	deal	account

The project has led to:

- implementation of the LDA model and collapsed Gibbs Sampling;
- method, based on perplexity computation, for assessing the convergence of collapsed Gibbs Sampling;
- analysis of execution time between different programming languages (Python, C++);
- qualitative analysis of results obtained on three different corpus (KOS, NIPS and ENRON).


KOS: perplexities (mean of cross−validation) for different # topics
perplexity
1550
1500
1450
1400
1350
1300
1250
1200
1150
1100
25
50
100
150
200
topics