Language and Document Analysis: Motivating Latent variable Models

Wray Buntine
National ICT Australia (NICTA)

MLSS, ANU, Jan., 2009

Part II

Problems and Methods

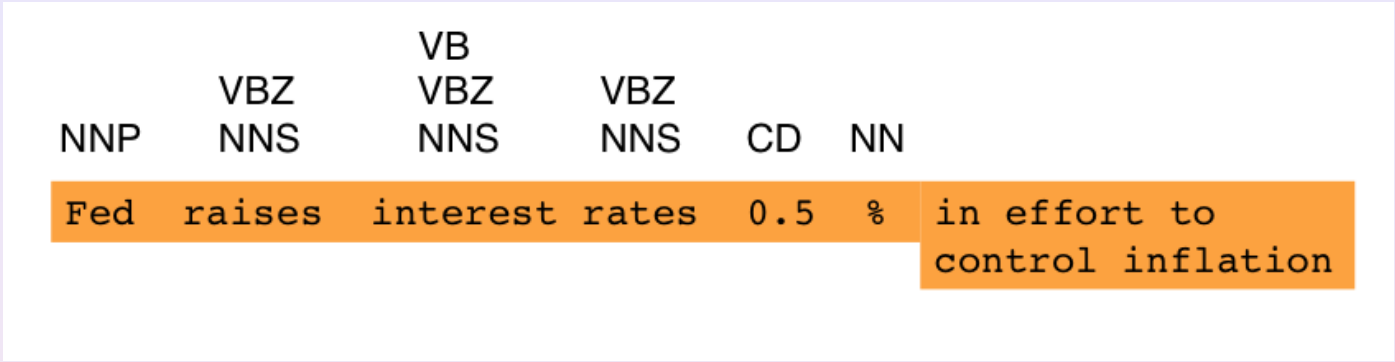
Outline

We review some key problems and key algorithms using latent variables.

Outline

We look at the Hidden Markov Model, because its an important base algorithm. We use it to introduce Conditional Random Fields, a recent high performance algorithm.

Parts of Speech, A Useful Example



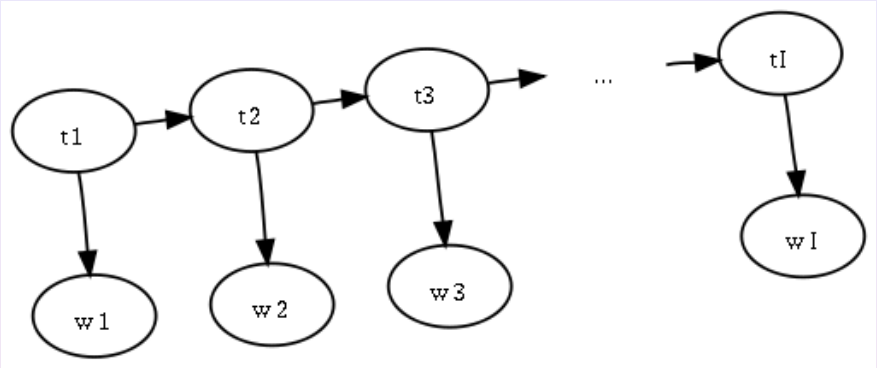
- A set of candidate POS exist for each word. taken from a dictionary or lexicon. Which is the right one in this sentence?
- Lets take some fully tagged data, where the truth is known, and use statistical learning.
- A standard notation for representing tags , in this example, is:

  `Fed/NNP raises/VBZ interest/NNS rates/NNS 0.5/CD %/% ...` *(in effort to control inflation.)*

- We use this to illustrate Markov models and HMMs.
  Reference: Manning and Schütze, chaps 9 and 10.

Outline

Markov Model with Known Tags



- There are I words. w_i = i-th word. t_i = tag for i-th word.
- Our 1st-order Markov model in the figure shows which variables depend on which.
- The $(i+1)$-th tag depends on the i-th tag. The i-th word depends on the i-th tag.
- Resultant formula for $p(t_1, t_2, t_3, ..., t_N, w_1, w_2, w_3, ..., w_N)$ is

$$p(t_1) \prod_{i=2,...,I} p(t_i|t_{i-1}) \prod_{i=1,...,I} p(w_i|t_i)$$

Fiitting Markov Model with Known Tags

- Have $p(t_1, t_2, t_3, ..., t_N, w_1, w_2, w_3, ..., w_N)$ is

$$p(t_1) \prod_{i=2,...,I} p(t_i|t_{i-1}) \prod_{i=1,...,I} p(w_i|t_i)$$

- Have K distinct tags and J distinct words.
- Use $p(t_i = k_1|t_{i-1} = k_2) = a_{k_2,k_1}$, $p(t_1 = k) = c_k$, $p(w_i = j|t_i = k) = b_{k,j}$.
- **a** and **b** are probability matrices whose columns sum to one.
- Collecting like terms

$$\prod_k c_k^{S_k} \prod_{k_1,k_2} a_{k_1,k_2}^{T_{k_1,k_2}} \prod_{k,j} b_{k,j}^{W_{k,j}}$$

where T_{k_1,k_2} is count of times tag k_2 follows tag k_1, and
$W_{k,j}$ is count of times tag k assigned to word j, and
S_k is count of times sentence starts with tag k.

Fiitting Markov Model with Known Tags, cont.

- Standard maximum likelihood methods apply, so these parameters **a** and **b** become their observed proportions:
  - a_{k_1,k_2} is proportion of tags of type k_2 when previous was k_1,
  - $b_{k,j}$ is proportion of words of type j when tag was k,
- Thus $a_{k_1,k_2} = \frac{T_{k_1,k_2}}{\sum_{k_2} T_{k_1,k_2}}$, $b_{k,j} = \frac{W_{k,j}}{\sum_j W_{k,j}}$, $c_k = \frac{S_k}{\sum_k S_k}$.
- Note we have many sentences in the training data, and each one has a fresh start, so c_k is estimating from all those initial tags in sentences.
- As is standard when dealing with frequencies, we can smooth these out by adding small amounts to the numerator and denominator to make all quantities non-zero.

Comments

- In practice, the naive estimation of **a** and **b** works poorly because we never have enough data. Most words occur infrequently, so we cannot get good tag statistics for them.
- Kupiec (1992) suggested grouping infrequent words together based on their pattern of candidate POS. This overcomes paucity of data with a reasonable compromise.
  - So "red" and "black" can both be NN or JJ, so they belong to the same *ambiguity class*.
  - Ambiguity classes not used for frequent words.
- Unknown words are also a problem. A first approximation is to assign unknown words with first capitals to NP.

Estimating Tags for New Text

- We now fix the Markov model parameters **a**, **b** and $\vec{c}$.
- We have a new sentence with I words $w_1, w_2, ..., w_I$. How do we estimate its tag set?
- We ignore the lexical contraints for now (*e.g.*, "interest" is VB, VBZ or NNS), and fold them in later.
- Task so described is:

$$\vec{t} \;=\; \text{argmax}_{\vec{t}}\; p\left(\vec{t}, \vec{w} \,|\, \mathbf{a}, \mathbf{b}, \vec{c}\right)$$

where the probability is as before.

Estimating Tags for New Text, cont.

Wish to solve

$$\text{argmax}_{\vec{t}}\ p(t_1) \prod_{i=2,\ldots,I} p(t_i|t_{i-1}) \prod_{i=1,\ldots,I} p(w_i|t_i)$$

The task is simplified by the fact that knowing the value for tag t_N splits the problem neatly into parts, so define

$$\begin{aligned} m(t_1) &= p(t_1) \\ m(t_N) &= \max_{t_1,\ldots,t_{N-1}|t_N}\ p(t_1) \prod_{i=2,\ldots,N} p(t_i|t_{i-1}) \prod_{i=1,\ldots,N-1} p(w_i|t_i) \end{aligned}$$

We get the recursion for $m(t_{N+1})$:

$$\begin{aligned} &= \max_{t_1,\ldots,t_N|t_{N+1}}\ p(t_1) \prod_{i=2,\ldots,N+1} p(t_i|t_{i-1}) \prod_{i=1,\ldots,N} p(w_i|t_i) \\ &= \max_{t_N|t_{N+1}} \max_{t_1,\ldots,t_{N-1}|t_N,t_{N+1}}\ p(t_1) \prod_{i=2,\ldots,N+1} p(t_i|t_{i-1}) \prod_{i=1,\ldots,N} p(w_i|t_i) \\ &= \max_{t_N}\ p(t_{N+1}|t_N)p(w_N|t_N)m(t_N) \end{aligned}$$

Estimating Tags for New Text, cont.

We apply this incrementally, building up a contingent solution from left to right. This is called the **Viterbi algorithm**, first developed in 1967.

1. Initialise $m(t_1)$, $m(t_1 = k) = c_k$.
2. For $i = 2, ..., I$, compute $m(t_i)$,

$$m(t_i = k_1) = \max_{k_2} (a_{k_2,k_1} b_{k_2,w_N} m(t_{i-1} = k_2))$$

   then store the backtrace, the k_2 that achieves maximum for each $t_i = k_1$.
3. At the end, I, find the maximum $t_I = \text{argmax}_k m(t_I = k)$, and chain through the backtraces to get the maximum sequence for $t_1, \ldots, t_I$.

This technique is an example of dynamic programming.

Comments

- What about lexical contraints, *e.g.*, our dictionary tells us that "interest" is either VB, VBZ or NNS?
- Thus $p(w_i = \text{'interest'} \mid t_i = \text{'JJS'}) = 0$.
- Thus we would like to enforce zeros in some entries of the **b** matrix.
- Likewise, with the ambiguity classes above, and with the individual words, we just assign some zero's to $b_{k,j}$ for j the index of the word.

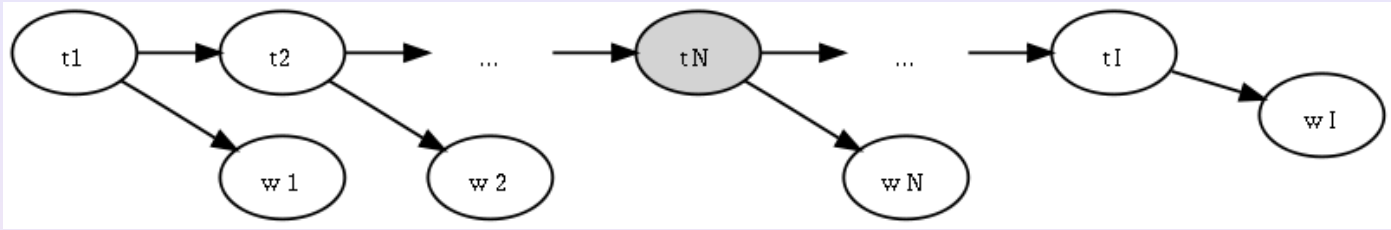
Estimating Tag Probabilities

- We again fix the Markov model parameters $\mathbf{a}$, $\mathbf{b}$ and $\vec{c}$.
- We have a new sentence with I words $w_1, w_2, ..., w_I$. We've got the most likely tag set using the Viterbi algorithm. What's the uncertainty here?
- Task can be described as: find the tag probabilities for each t_N.

$$p(t_N|\vec{w}) \propto \sum_{\vec{t}/t_N} p\left(\vec{t}, \vec{w} \,|\, \mathbf{a}, \mathbf{b}, \vec{c}\right)$$

where the probability is as before.

Estimating Tag Probabilities, cont.



Wish to compute $p(t_N|\vec{w})$, got by normalising

$$p(t_N, \vec{w}) = \sum_{\vec{t}/t_N} \left(p(t_1) \prod_{i=2,\ldots,I} p(t_i|t_{i-1}) \prod_{i=1,\ldots,I} p(w_i|t_i) \right)$$

Note we have:

$$p(t_N|w_1, \ldots, w_{N-1}) = \sum_{t_1,\ldots,t_{N-1}} \left(p(t_1) \prod_{i=2,\ldots,N} p(t_i|t_{i-1}) \prod_{i=1,\ldots,N-1} p(w_i|t_i) \right)$$

$$p(w_{N+1}, \ldots, w_I|t_N) = \sum_{t_{N+1},\ldots,t_I} \left(\prod_{i=N+1,\ldots,I} p(t_i|t_{i-1}) \prod_{i=N+1,\ldots,I} p(w_i|t_i) \right)$$

$$\textit{Thus} \quad p(t_N, \vec{w}) = p(t_N|w_1, \ldots, w_{N-1}) p(w_{N+1}, \ldots, w_I|t_N) p(w_N|t_N)$$

Estimating Tag Probabilities, cont.

The quantities $p(t_N|w_1,...,w_{N-1})$ and $p(w_{N+1},...,w_I|t_N)$ are traditionally called $\alpha(t_N)$ and $\beta(t_N)$ respectively.

As with the Viterbi, a recursion exists:

$$p(t_N|w_1,...,w_{N-1}) = \sum_{t_{N-1}} p(t_N|t_{N-1})p(w_{N-1}|t_{N-1})p(t_{N-1}|w_1,...,w_{N-2})$$

$$p(w_{N+1},...,w_I|t_N) = \sum_{t_{N+1}} p(t_{N+1}|t_N)p(w_{N+1}|t_{N+1})p(w_{N+2},...,w_I|t_{N+1})$$

Compute the first with a forward pass in N, compute the second with a backward pass in N. Hence computing these probabilities is called the *Forward-Backward* algorithm. Complexity is $O(I\,K^2)$.

$$\alpha_N(k_1) = \sum_{k_2} a_{k_2,k_1} b_{k_2,w_{N-1}} \alpha_{N-1}(k_2)$$

$$\beta_N(k_1) = \sum_{k_2} a_{k_1,k_2} b_{k_2,w_{N+1}} \beta_{N+1}(k_2)$$

$$\alpha_1(k) = c_k \qquad \beta_I(k) = 1$$

Outline

Fitting with Unknown Tags

- We don't always have a large quantity of text tagged with POS. So we would like to try and improve the estimates of the model using untagged or partially tagged data.
- So the problem becomes, estimate $\mathbf{a}$, $\mathbf{b}$ and $\vec{c}$ given the sequence $w_1, w_2, ..., w_l$ but no tags.
- The case with partial tags can be folded in later.
- This problem, where the tags are unknown initially is called a *hidden Markov model* (HMM).

A Little Bit of Magic

- We will use some probability function $q(\vec{t})$ in our solution as a device. This represents *some* valid probability over the tags. NB. it can be represented by a large parameter vector.
- For brevity, refer to **a**, **b** and $\vec{c}$ by a single parameter vector $\vec{\theta}$.
- Consider the function $Q(\vec{\theta}, q())$ given by

$$\begin{aligned} &= \log p(\vec{w}|\vec{\theta}) - KL\left(q(\vec{t}) \,||\, p(\vec{t}|\vec{w},\vec{\theta})\right) \\ &= E_{q(\vec{t})}\left(\log p(\vec{t},\vec{w}|\vec{\theta})\right) + I(q(\vec{t})) \end{aligned}$$

- A simple expansion of $KL()$ and $I()$ shows the two forms are equal.

A Little Bit of Magic, cont.

- Consider the function $Q(\vec{\theta}, q())$ given by

$$\begin{aligned} &= \log p(\vec{w}|\vec{\theta}) - KL\left(q(\vec{t}) \,||\, p(\vec{t}|\vec{w},\vec{\theta})\right) \\ &= E_{q(\vec{t})}\left(\log p(\vec{t},\vec{w}|\vec{\theta})\right) + I(q(\vec{t})) \end{aligned}$$

- Maximise this w.r.t. $\vec{\theta}$ and $q()$ jointly.
- By the first equation, this holds when $q(\vec{t}) = p(\vec{t}|\vec{w},\vec{\theta})$, and then $Q(\vec{\theta}, q()) = \log p(\vec{w}|\vec{\theta})$.
- By the second equation, this holds if we solve:

$$\text{argmax}_{\vec{\theta}}\ E_{q(\vec{t})}\left(\log p(\vec{t},\vec{w}|\vec{\theta})\right) \ .$$

- Thus, iterating these two steps will achieve the maximum likelihood solution $\text{argmax}_{\vec{\theta}}\ \log p(\vec{w}|\vec{\theta})$.

Fitting with Unknown Tags, cont.

The "conceptual" algorithm is to repeatedly re-estimate $\vec{\theta}$.

1. Construct the intermediate distribution $q(\vec{t}) = p(\vec{t}|\vec{w}, \vec{\theta})$ from the current $\vec{\theta}$.

   $\longrightarrow$ This maximizes $\left(\log p(\vec{w}|\vec{\theta}) - KL\left(q(\vec{t}) \,||\, p(\vec{t}|\vec{w}, \vec{\theta})\right)\right)$ w.r.t. $q()$.

2. Use this to evaluate $C(\vec{\theta}) = E_{q(\vec{t})}\left(\log p(\vec{t}, \vec{w}|\vec{\theta})\right)$.

3. Now re-maximise $\vec{\theta}' = \text{argmax}_{\vec{\theta}}\, C(\vec{\theta})$.

   $\longrightarrow$ This maximizes $\left(E_{q(\vec{t})}\left(\log p(\vec{t}, \vec{w}|\vec{\theta})\right) + I(q(\vec{t}))\right)$ w.r.t. $\vec{\theta}$.

This is called the *Expectation-Maximization algorithm*, or EM for short. It works efficiently when can get the formula for the steps of the conceptual algorithm.

Revision

We wish to evaluate $E_{q(\vec{t})}\left(\log p(\vec{t}, \vec{w}|\vec{\theta})\right)$, where:

$$p(\vec{t}, \vec{w}|\vec{\theta}) = \prod_k c_k^{S_k} \prod_{k_1,k_2} a_{k_1,k_2}^{T_{k_1,k_2}} \prod_{k,j} b_{k,j}^{W_{k,j}} ,$$

where T_{k_1,k_2} is count of times tag k_2 follows tag k_1, and
$W_{k,j}$ is count of times tag k assigned to word j, and
S_k is count of times sentence starts with tag k.

T_{k_1,k_2}, $W_{k,j}$ and S_k are statistics for the tags $\vec{t}$ given words $\vec{w}$.

Fitting with Unknown Tags, cont.

Linearity of expectation gives $E_{q(\vec{t})}\left(\log p(\vec{t}, \vec{w}|\vec{\theta})\right)$

$$
\begin{aligned}
&= E_{q(\vec{t})}\left(\sum_k \log c_k^{S_k} + \sum_{k_1,k_2} \log a_{k_1,k_2}^{T_{k_1,k_2}} + \sum_{k,j} \log b_{k,j}^{W_{k,j}}\right) \\
&= \sum_k E_{q(\vec{t})}(S_k) \log c_k + \sum_{k_1,k_2} E_{q(\vec{t})}(T_{k_1,k_2}) \log a_{k_1,k_2} + \sum_{k,j} E_{q(\vec{t})}(W_{k,j}) \log b_{k,j} \ ,
\end{aligned}
$$

where the expected values are given by:

$$
\begin{aligned}
E_{q(\vec{t})}(S_k) &= p(t_0{=}k|q(\vec{t})) \\
E_{q(\vec{t})}(T_{k_1,k_2}) &= \sum_i p(t_{i+1}{=}k_2|t_i{=}k_1, q(\vec{t})) \\
E_{q(\vec{t})}(W_{k,j}) &= \sum_i 1_{w_i=j} p(t_i{=}k|q(\vec{t}))
\end{aligned}
$$

Fitting with Unknown Tags, cont.

Maximising

$$\sum_k E_{q(\vec{t})}(S_k) \log c_k + \sum_{k_1,k_2} E_{q(\vec{t})}(T_{k_1,k_2}) \log a_{k_1,k_2} + \sum_{k,j} E_{q(\vec{t})}(W_{k,j}) \log b_{k,j}$$

w.r.t. the probability matrices and vectors $\mathbf{a}$, $\mathbf{b}$ and $\vec{c}$ is a standard constrained optimisation problem. Remember the columns of $\mathbf{a}$, $\mathbf{b}$ must add to one.

The solution is:

$$\begin{aligned} c_k &\propto E_{q(\vec{t})}(S_k) \\ a_{k_1,k_2} &\propto E_{q(\vec{t})}(T_{k_1,k_2}) \\ b_{k,j} &\propto E_{q(\vec{t})}(W_{k,j}) \end{aligned}$$

Baum-Welch Algorithm

Putting it all together.

1. From the current solution for **a**, **b** and $\vec{c}$, perform the Forward-Backward algorithm to compute $\alpha_N(\cdot)$ and $\beta_N(\cdot)$.
2. From these, compute

$$p(t_N{=}k|q()) \;\propto\; \alpha_N(k)\beta_N(k)b_{k,W_N} \;,$$

$$p(t_N{=}k_2|t_{N-1}{=}k_1, q()) \;\propto\; \alpha_{N-1}(k_1)\beta_N(k_2)b_{k_1,W_{N-1}}b_{k_2,W_N}a_{k_1,k_2} \;.$$

3. Hence compute $E_{q(\vec{t})}(S_k)$, $E_{q(\vec{t})}(T_{k_1,k_2})$ and $E_{q(\vec{t})}(W_{k,j})$ using formula on previous page.
4. Now maximise for **a**, **b** and $\vec{c}$ using the proportions on the previous page.

Comments

- This is called the Baum-Welch algorithm, after the original inventors. It is an instance of the so-called EM algorithm.
- Unfortunately this HMM training doesn't work too well for the POS problem. Although, it was for a long time the best method for speech to text recognition.
- Perhaps the poor performance on POS tagging is because we are fitting a joint model $p(\vec{t}, \vec{x}|\vec{\theta})$ rather than a conditional model $p(\vec{t}|\vec{x}, \vec{\theta})$.
- So lets investigate conditional models.

Conditional Fitting with Unknown Tags

- So the problem is to estimate a model for $\vec{t}$ given the sequence $w_1, w_2, ..., w_I$ but no tags.
- We no longer have $p(w_i|t_i)$, rather we want a discriminative model, something like $p(t_i|w_i)$, but also $p(t_i|t_{i-1})$,
- One approach, called the conditional random field (CRF) is to fold them in together to get:

$$p(\vec{t}\,|\,\vec{w}, \mathbf{a}, \mathbf{b}, \vec{c}) \;\propto\; \exp\left(\sum_i a_{t_{i-1},t_i} + \sum_i b_{t_i,w_i} + \sum_i c_{t_i}\right) .$$

- Compare this conditional model with our HMM model, which can be manipulated to

$$p(\vec{t}, \vec{w}|\mathbf{a}, \mathbf{b}, \vec{c}) \;=\; \exp\left(\sum_i a_{t_{i-1},t_i} + \sum_i b_{t_i,w_i} + \sum_i c_{t_i}\right) .$$

Conditional Fitting with Known Tags, cont.

- The so-called *conditional random field* has:

$$p(\vec{t} \,|\, \vec{w}, \mathbf{a}, \mathbf{b}, \vec{c}) \;\propto\; \exp\left(\sum_i a_{t_{i-1},t_i} + \sum_i b_{t_i,w_i} + \sum_i c_{t_i}\right)$$

- We need a normalising constant, Z, a function of $\mathbf{a}$, $\mathbf{b}$ and $\vec{c}$.

$$Z \;=\; \sum_{\vec{t}} p(\vec{t} \,|\, \vec{w}, \mathbf{a}, \mathbf{b}, \vec{c})$$

- Compute this incrementally, rather like a forward pass of the Forward-Backward algorithm.

$$\begin{aligned} Z_1(t_1) &= 1 \\ Z_N(t_N) &= \sum_{t_{N-1}} Z_{N-1}(t_{N-1}) \exp\left(a_{t_{N-1},t_N} + b_{t_{N-1},w_{N-1}} + c_{t_{N-1}}\right) \\ Z &= \sum_{t_N} Z_N(t_N) \exp\left(b_{t_N,w_N} + c_{t_N}\right) \end{aligned}$$

Conditional Fitting with Known Tags, cont.

We have to use gradient based algorithms to fit this as there are no closed forms.

- Lets look at the likelihood to maximise $\log p(\vec{t}|\vec{w},\vec{\theta})$.

$$\sum_k S_k c_k + \sum_{k_1,k_2} T_{k_1,k_2} a_{k_1,k_2} + \sum_{k,j} W_{k,j} b_{k,j} - \log Z$$

- Note **a**, **b** and $\vec{c}$ are no longer probability matrices and vectors.
- Now it happens that

$$\begin{aligned} \frac{\partial \log Z}{\partial a_{k_1,k_2}} &= E_{p(\vec{t}|\vec{w},a,b,\vec{c})}(T_{k_1,k_2}) \ , \\ \frac{\partial \log Z}{\partial b_{k,j}} &= E_{p(\vec{t}|\vec{w},a,b,\vec{c})}(W_{k,j}) \ . \end{aligned}$$

  These expected values can be computed by a variant of the forward-backward algorithm, as before.

- Thus we have all the derivatives of the likelihood.

Comments

- Training slower than for a HMM.
- Conditional training with some unknown tags also works, but is more complicated again.
- In principle. you can now use any features, not just the words $\vec{w}$. People use:
  - capitalisation, all-caps, use of non-alphabetic letters,
  - presence of prefixes and suffixes,
  - properties of surrounding words,
  - match of words to different gazetteers.
- In this case, the performance is very dependent on the choice of features!

Outline

Methods for discovering hiiden components or topics in semi-structured data.
Reference: Buntine and Jakulin, 2006.

Outline

Motivation

- Web industry players are exploring the use of topic models for text. *e.g.*, Microsoft, Yahoo, various startups.
- Large amounts of text in different context available (blogs, news, corporate, Wikipedia, language, ...).
- Current processing performance is of the order of one million documents on a multi-core system in a few days.

Motivation

- We start with a collection of documents in some area.
- We'd like to discover the topics in the collection automatically, using *unsupervised learning*.
- A document is modelled as having multiple topics, for instance one sports article can have three component topics: Argentina, Soccer, and Crowd Behaviour.
- A topic is modelled as a set of words that frequently occur together.

Viewing Topics at the Word Level (Blei, Ng, and Jordan, 2003)

"Arts"	"Budgets"	"Children"	"Education"
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

Figure 8: An example article from the AP corpus. Each color codes a different factor from which the word is putatively generated.

Example: Topics in the Wikipedia

- We take 1 million documents from the Wikipedia, and tokenise the text in each document, without linguistic processing.
- This yields about half a gigabyte of binary data.
- We train the topic models and then look at the topics.

Example Topic: Mythology

NOUNS					
mythology	0.03337	God	0.02048	name	0.014747
goddess	0.012911	spirit	0.012639	legend	0.0087992
myth	0.0070882	demons	0.006807	Sun	0.0060099
Temple	0.0054717	deity	0.0054247	Bull	0.0051629
Dragon	0.0051379	Maya	0.0051243	King	0.00512
Sea	0.0049453	Norse	0.0044707	horse	0.0044592
symbol	0.0042196	animals	0.0040112	fire	0.0039879
hero	0.0038755	Romans	0.0038696	Apollo	0.0037588
VERBS					
called	0.034078	said	0.031081	see	0.029521
given	0.0269	associated	0.024591	according	0.021724
represented	0.020964	known	0.018896	could	0.017499
made	0.016952	depicted	0.01524	appeared	0.014662
ADJECTIVES					
Greek	0.091163	ancient	0.055393	great	0.02853
Egyptian	0.028071	Roman	0.025783	sacred	0.020446

Historical Background

Long history of component models before the discrete topics models we consider:

- Principal Components Analysis (PCA), dimensionality reduction tool, invented by Karl Pearson in 1901, theoretical relationship to least squares and Gaussians.
- Independent Components Analysis (ICA), invented by Herault and Jutten in 1986, for *blind source separation* of image and signal data, usually used with PCA.
- Latent Semantic Indexing (LSI), intended for text in IR, but of mixed benefit, and difficult to interpret, a variant of PCA.

Gaussian and least squares models fail for the smaller counts data we are considering. Need Poisson or multinomial modelling instead.

Discrete Topic Models, a Short History

- Soft clustering, "grade of membership", Woodbury & Manton, 1982.
- Admixture modelling in statistics, 1980s.
- Hidden facets in image interpretation, Non-negative Matrix Factorization (NMF), Seung and Lee, 1999.
- Probabilistic Latent Semantic Analysis (PLSI), topics in text, Hofmann, 1999.
- Admixture modelling, fully Bayesian, population structure from genotype data, Pritchard, Stephens and Donnelly, 2000.
- Latent Dirichlet Allocation (LDA) Blei, Ng and Jordan, 2001. Variant of Pritchard *et al.* Introduced mean-field algorithm.
- Collapsed Gibbs sampler, Griffiths and Steyvers, 2004.
- Gamma-Poisson model (GaP), Canny 2004 (extension of NMF).

... variants, extensions, adaptations, ..., 2001-2008

Bag of words to represent text

A page out of Dr. Zeuss's *The Cat in The Hat*:

> *So, as fast as I could, I went after my net. And I said, "With my net I can bet them I bet, I bet, with my net, I can get those Things yet!"*

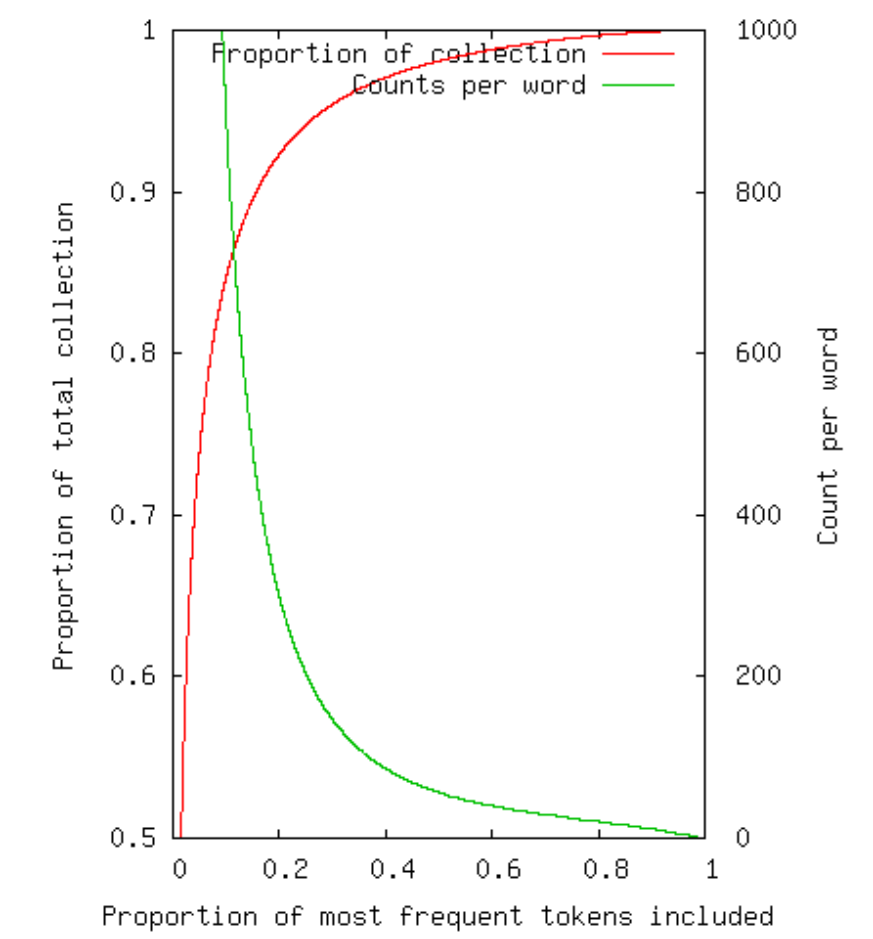
In the *bag of words* representation as *word* (*count*):

> *after(1) and(1) as(2) bet(3) can(2) could(1) fast(1) get(1) I(7) my(3) net(3) said(1) so(1) them(1) things(1) those(1) went(1) with(2) yet(1) .*

Notes:

- For the Reuters RCV1 collection from 2000: $I \approx 800k$ documents, $J \approx 400k$ different words (excluding those occurring few times), $S \approx 300M$ words total.
- Represent as sparse matrix/vector form with integer entries.
- Compresses to about 2 bytes per token (*e.g.* $2S$ bytes) total storage.

Document-word tradeoffs



Data from NY Times collection from UCI.

- Deleting about 50% of the most infrequent words from the dictionary decreases the collection size by only about 3%.
- We can train on a subset of the dictionary as a way of boot-strapping.
- Shows that compression of various word matrices and vectors can be significant.
- Should also ignore words occurring in, say, 30% or more of documents as "stop" words.

Issues in text representation

- The basic semantic units in text are not words but, most commonly, compound words.
  - *e.g.*, "New York Times", "George Bush"
  - most common are single words.
  - occasionally compound words are *not* contiguous.
- Web pages full of "cruft", HTML junk, adverts, company fluff, navigation aids, boilerplate, ...
- Different "styles" of topics exists:

  genre: *e.g.*, product page, blog, news, corporate info.,
  library style categorisation: as done by Dewey Decimal, and DMOZ
  opinion and sentiment: *e.g.*, positive, anti-Microsoft, "green", ...

Outline

Basic Model

- Everything tokenised, so have I documents, J words in the dictionary, K different topics/components.
- The model of how frequent words are for a topic given by the *topic by word* matrix of proportions, Θ of dimension $J \times K$.
- The model of how topics are distributed in a given document given by a Dirichlet of dimension K with parameters $\vec{\alpha}$.
- For a given document i, we'll sample the topics proportions, a *latent or hidden* variable $\vec{m}_i$ as

$$\vec{m}_i \sim \text{Dirichlet}_K(\vec{\alpha})$$

- Words in a document i generated independently, proportion given by J-dimensional vector $\vec{m}_i^\dagger \Theta$. For sequence $l = 1, ..., L$

$$p(j_l \,|\, \Theta, \vec{m}_i) = \sum_k m_{i,k} \theta_{k,j_l} \ .$$

Basic Model

The model has the following *parameters*:

K: number of topics,

$\vec{\alpha}$: used to generate topics for each document,

Θ: word proportions for each topic.

The *sampling model* acts as follows:

1. For each document indexed by i:
   1. Generate the topic proportions for the document $\vec{m}_i \sim \text{Dirichlet}_K(\vec{\alpha})$.
   2. For each word indexed by l in the document i:
      1. generate the topic of the word $k_l \sim \text{Discrete}_K(\vec{m}_i)$,
      2. take the k_l-th column from Θ, which is $\vec{\theta}_{k_l}$, generate the word $j_l \sim \text{Discrete}_J(\vec{\theta}_{k_l})$.

Each document has *hidden (or latent) variables* $\vec{m}_i$ and $\vec{k}_i$.

Revision

The K-dimensional Dirichlet distribution is a function on a proportions $\vec{m}$, of the form

$$p(\vec{m} \,|\, \vec{\alpha}, K, \text{Dirichlet}) \;=\; \frac{1}{Z_K(\vec{\alpha})} \prod_{k \in Topics} m_k^{\alpha_k - 1} \;.$$

The normalising constant $Z_K(\vec{\alpha})$ evaluates as

$$\prod_k \Gamma(\alpha_k) \Big/ \Gamma\left(\sum_k \alpha_k\right)$$

Means are given by

$$\mathrm{E}(m_k) \;=\; \alpha_k \Big/ \sum_k \alpha_k$$

Document Likelihoods

The likelihood including all the latent variables, $p(\vec{j}_i, \vec{k}_i, \vec{m}_i \,|\, \text{for doc } i, \Theta, \alpha)$:

$$\left(\frac{1}{Z_K(\vec{\alpha})} \prod_{k \in Topics} m_{i,k}^{\alpha_k - 1} \right) \prod_{l \in WordSequence_i} m_{i,k_{i,l}} \theta_{k_{i,l}, j_{i,l}} \ .$$

Marginalising out the latent topic assignments, $\vec{k}_i$, giving $p(\vec{j}_i, \vec{m}_i \,|\, \text{for doc } i, \Theta, \alpha)$:

$$\left(\frac{1}{Z_K(\vec{\alpha})} \prod_{k \in Topics} m_{i,k}^{\alpha_k - 1} \right) \prod_{l \in WordSequence_i} \sum_{k \in Topics} m_{i,k} \theta_{k, j_{i,l}} \ .$$

Marginalising out instead the topic proportions $\vec{m}_i$, giving $p(\vec{j}_i, \vec{k}_i \,|\, \text{for doc } i, \Theta, \alpha)$, where $C_{i,k}$ is the count of topic k in document i,

$$\frac{Z_K(\vec{\alpha} + \vec{C}_i)}{Z_K(\vec{\alpha})} \prod_{l \in WordSequence_i} \theta_{k_{i,l}, j_{i,l}} \ .$$

Estimating Test Likelihoods

- The likelihood we would like to report in testing is, $p(\vec{j}_i \,|\, \text{for doc } i, \Theta, \alpha)$. We have no computable form for this.
- The previous likelihoods are almost certainly bad over-estimates, unless the latent/hidden variables used in evaluating them or sampled uniformly, in which case they are very poor estimates and useless.
- If we sample the topic assignments $\vec{k}_i$ proportionally too $p(\vec{j}, \vec{k} \,|\, \text{for doc } i, \Theta, \alpha)$, the document likelihood can be approximated as

$$\frac{1}{N} \Bigg/ \sum_{n=1}^{N} \frac{1}{\prod_{l \in WordSequence_i} \theta_{k_{n,l}, j_l}}$$

where we have N sample vectors $\vec{k}_n$.

See (Carlin and Chib, 1995).

Variational EM Algorithm: Rough Outline

Seeks to maximise the likelihood, $p(\vec{j}_i, \vec{m}_i \text{ for } i \in Docs \,|\Theta, \alpha)$, where $\vec{j}_i$ are the words for a document, $Z_k()$ is Dirichlet normaliser:

$$\prod_{i \in Docs} \left(\frac{1}{Z_K(\vec{\alpha})} \prod_{k \in Topics} m_{i,k}^{\alpha_k - 1} \right) \left(\prod_{l \in WordSequence} \sum_{k \in Topics} m_{i,k} \theta_{k,j_{i,l}} \right) .$$

Typically consists of a few hundred cycles in the form

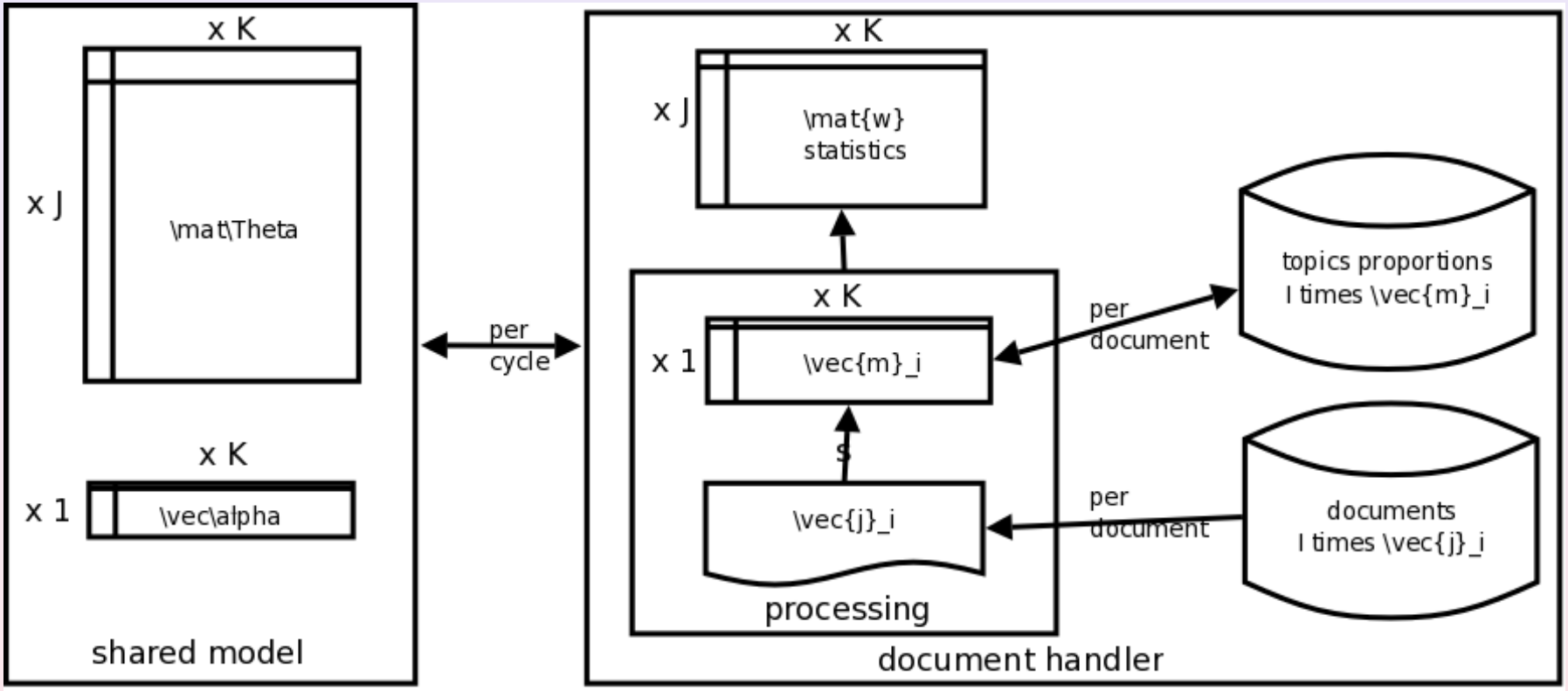
1. For each document i, re-estimate/improve values for $\vec{m}_i$, based on the factored approximation

$$\frac{1}{Z_K(\vec{\alpha})} \prod_{k \in Topics} m_{i,k}^{\alpha_k - 1} \left(\prod_{l \in WordSequence} \prod_{k \in Topics} m_{i,k}^{\mu_{k,l}} \right) .$$

2. Re-assign values for Θ based on statistics collected in step (1), based on the factored approximation

$$\prod_i \prod_l \prod_k \theta_{k,j_{i,l}}^{m_{i,k}} .$$

Parallel Variational EM



Parallel Variational EM, notes

- Distribute documents to different document handlers.
- The documents $\vec{j}_i$, and the document proportions $\vec{m}_i$ can be streamed, so are not a significant memory cost.
- $\vec{m}_i$ will need to be compressed when K is large.
- Need to communicate Θ and α with each major cycle: collect statistics, then distribute update; efficient primitives should be used for communiciation.

Collapsed Gibbs Algorithm: Derivation

Take the likelihood, $p(\vec{j}_i, \vec{m}_i \text{ for } i \in Docs \,|\Theta, \alpha)$:

$$\prod_{i \in Docs} \left(\frac{1}{Z_K(\vec{\alpha})} \prod_{k \in Topics} m_{i,k}^{\alpha_k - 1} \right) \left(\prod_{l \in WordSequence_i} \sum_{k \in Topics} m_{i,k} \theta_{k,j_{i,l}} \right) .$$

Introduce the topics per word, $p(\vec{j}_i, \vec{k}_i, \vec{m}_i \text{ for } i \in Docs \,|\Theta, \alpha)$

$$\prod_{i \in Docs} \left(\frac{1}{Z_K(\vec{\alpha})} \prod_{k \in Topics} m_{i,k}^{\alpha_k - 1} \right) \left(\prod_{l \in WordSequence_i} m_{i,k_{i,l}} \theta_{k_{i,l},j_{i,l}} \right) .$$

Collect terms in Θ and $\vec{m}_i$, with statistics $\vec{W}$ and $\vec{C}_i$ respectively, and integrate/marginalise $\vec{m}_i$, giving $p(\vec{j}_i, \vec{k}_i \text{ for } i \in Docs \,|\Theta, \alpha)$

$$\prod_{k \in Topics} \prod_{j \in Words} \theta_{k,j}^{W_{k,j}} \prod_{i \in Docs} \frac{Z_K(\vec{\alpha} + \vec{C}_i)}{Z_K(\vec{\alpha})} .$$

Collapsed Gibbs Algorithm: Derivation, cont.

Finally, integrate/marginalise Θ (by adding prior for Θ of $\vec{\gamma}$) $p(\vec{j}_i, \vec{k}_i \text{ for } i \in Docs \,|\vec{\alpha}, \vec{\gamma})$

$$\prod_{k \in Topics} \frac{Z_J(\vec{\gamma} + \vec{W}_k)}{Z_J(\vec{\gamma})} \prod_{i \in Docs} \frac{Z_K(\vec{\alpha} + \vec{C}_i)}{Z_K(\vec{\alpha})} \,.$$

Substituting the normalising constant $Z(\cdot)$ yields

$$\prod_{k \in Topics} \left(\frac{\Gamma\left(\sum_j \gamma_j\right)}{\Gamma\left(\sum_j (\gamma_j + W_{k,j})\right)} \prod_{j \in Words} \frac{\Gamma(\gamma_j + W_{k,j})}{\Gamma(\gamma_j)} \right)$$

$$\prod_{i \in Docs} \left(\frac{\Gamma\left(\sum_k \alpha_k\right)}{\Gamma\left(\sum_k (\alpha_k + C_{i,k})\right)} \prod_{k \in Topics} \frac{\Gamma(\alpha_k + C_{i,k})}{\Gamma(\alpha_k)} \right)$$

Collapsed Gibbs Algorithm: Rough Outline

Probability $p(\vec{k}_i \text{ for } i \in Docs \,|\vec{j}_i \text{ for } i \in Docs, \vec{\alpha}, \vec{\gamma})$

$$\propto \prod_{k \in Topics} \left(\frac{\Gamma\left(\sum_j \gamma_j\right)}{\Gamma\left(\sum_j (\gamma_j + W_{k,j})\right)} \prod_{j \in Words} \frac{\Gamma(\gamma_j + W_{k,j})}{\Gamma(\gamma_j)} \right)$$

$$\prod_{i \in Docs} \left(\frac{\Gamma\left(\sum_k \alpha_k\right)}{\Gamma\left(\sum_k (\alpha_k + C_{i,k})\right)} \prod_{k \in Topics} \frac{\Gamma(\alpha_k + C_{i,k})}{\Gamma(\alpha_k)} \right)$$

Change the topic assignment for one word, $k_{i,l}$, gives simple product formula for a Gibbs update on $k_{i,l}$. See Griffiths and Steyvers 2004.

$$p(k_{i,l} = k \,|\, j_{i,l} = j, \vec{W}, \vec{C}, \vec{\alpha}, \vec{\gamma}) \;\propto\; (C_{i,k} + \alpha_k) \frac{W_{k,j} + \gamma_j}{\sum_j (W_{k,j} + \gamma_j)}$$

where $\vec{C}_i$ is the topic totals for document i, and $\vec{W}$ is the topic totals by word.

Collapsed Gibbs Algorithm: Rough Outline, cont.

The formula for a Gibbs update on $k_{i,l}$:

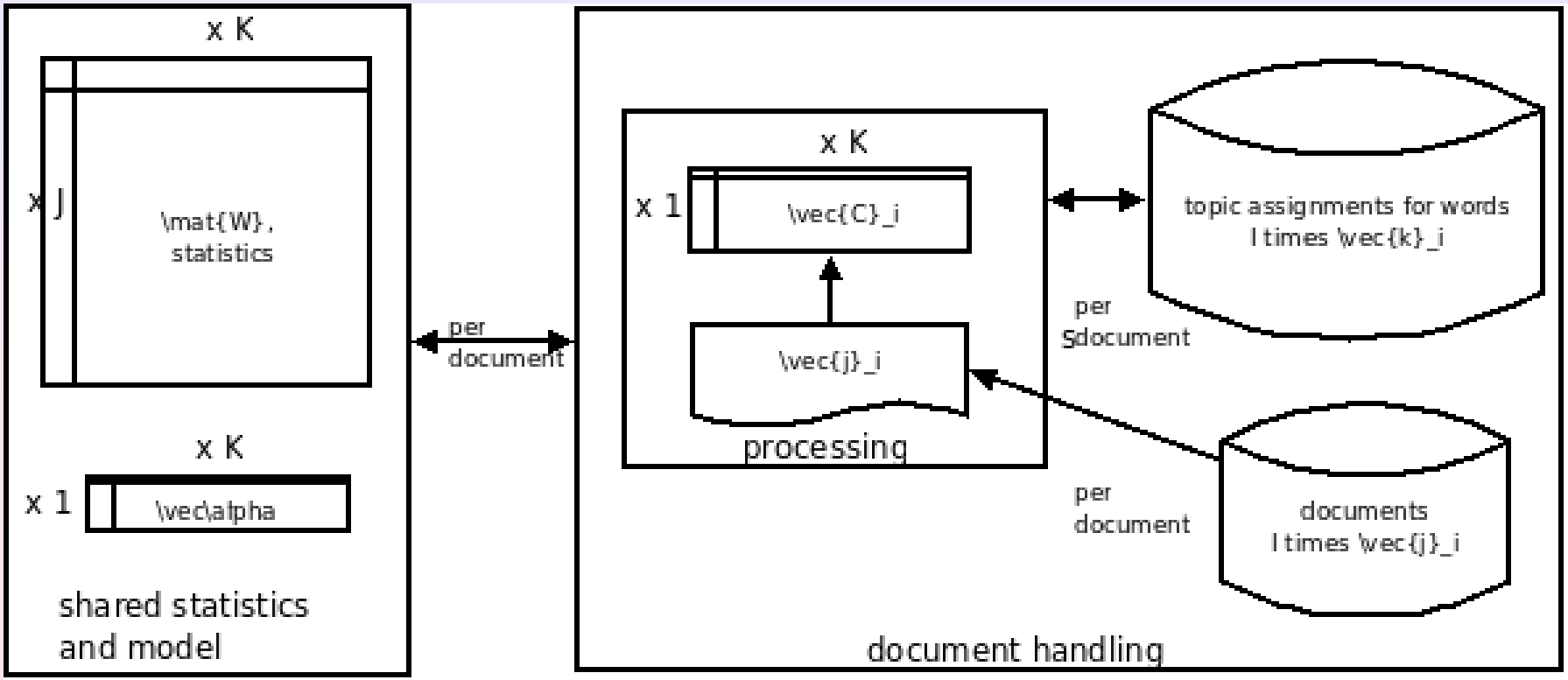
$$p(k_{i,l} = k \,|\, j_{i,l} = j, \vec{W}, \vec{C}, \vec{\alpha}, \vec{\gamma}) \;\propto\; (C_{i,k} + \alpha_k) \frac{W_{k,j} + \gamma_j}{\sum_j (W_{k,j} + \gamma_j)}$$

where $\vec{C}_i$ is the topic totals for document i, and $\vec{W}$ is the topic totals by word.

Algorithm consists of, say, thousand cycles in the form:

1. For each document i,
   1. Recompute topic totals $\vec{C}_i$ from stored topic assignments $\vec{k}_i$.
   2. For sequence $l = 1, ..., L$ in document,
      1. for word $j_{i,l}$, re-sample its topic assignment $k_{i,l}$ using statistics $\vec{W}$ and $\vec{C}_i$,
      2. update $\vec{W}$ and $\vec{C}_i$.

Parallel Collapsed Gibbs EM



Parallel Collapsed Gibbs, notes

- Distribute documents to different document handlers.
- The documents $\vec{j}_i$, and their topic assignment $\vec{k}_i$ can be streamed, again. Both about the same size.
- Need to update statistics W continuously! Best batch it, find the difference, compress, and communicate.
- W compressible by factor of 2-20, or more if many document handlers involved.

Various papers from UCI group on distributed LDA, and the `ParallelTopicModel.java` code of Mallet, by Mimno and McCallum.

Comments

- Many of the published models, NMF with K-L metric, GaP, LDA and PLSI are all variations of one another if we ignore hyperparameters, and the statistical and optimisation methods used.
  NB. Bregman divergence variations also exist.
- Different kinds of algorithms used: variational EM, maximum likelihood, Gibbs sampling, collapsed Gibbs sampling, ...
- Lots of extensions exist:
  - using N-th (usually 2nd) order Markov models on words,
  - hierarchical extensions,
  - correlated topics, *e.g.*, Pachinko,
  - sparse matrices,
  - time dependent or otherwise conditional topics.