Bayesian unsupervised frame learning from text
Data Analysis Project, draft report

Brendan O'Connor
Machine Learning Department, Carnegie Mellon University

February 22nd, 2012

1 Introduction: Frame induction for corpus analysis

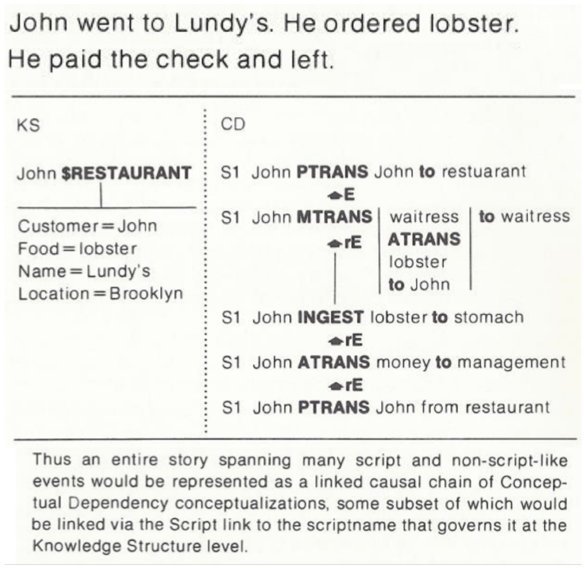
The use of computational methods to explore research questions in the social sciences has boomed over the past several years, as the volume of data capturing human communication—e.g. massive social media data streams—has risen to match the ambitious goal of understanding the behaviors of people and society (Lazer et al., 2009). Automated content analysis of text draws on techniques developed in natural language processing, information retrieval, and machine learning, and has been applied to political science, economics, psychology, and many other areas. Statistical summaries of language use in a large corpus can give insight into the opinions, beliefs, and situations communicated in the texts. In O'Connor et al. (2011), we overview this field and note that while state-of-the-art techniques have great variety in statistical and computational complexity (as in our own work (O'Connor et al., 2010a; Eisenstein et al., 2010; O'Connor et al., 2010b; Yogatama et al., 2011; Bamman et al., 2011), as well as dozens of other examples), in nearly all cases, researchers restrict their analysis to word and phrase frequencies. But this falls far short of the insight an expert can achieve when reading a text—in news, social media, literature, religious texts, and other corpora, the texts can encode beliefs, opinions, events, arguments, and overarching narrative themes. Can we automatically extract these from raw text?

As a first step, in this project we focus on *frame and script learning*—inducing a system of typical events, actors, and actors' roles, that tend to co-occur within documents. We use a Bayesian generative modeling approach, building on previous successes in unsupervised learning of latent semantic structure in documents and language.

2 Frames and scripts

Classic work in artificial intelligence and linguistics proposes that knowledge has a structured representation of *frames*, *schemata*, or *scripts* (Minsky 1974, Fillmore 1982, Rumelhart 1980, Schank and Abelson 1977). Figure 1a shows an example from Schank and Abelson, where an utterance evokes two levels of analysis: a deep "knowledge structure" or "script," that describes the typical aspects of visiting a restaurant; and a lower-level "conceptual dependency" structure, that describes a series of individual relations and events.

The area of template-filling information extraction, typified by the Message Understanding Conferences in the late 80's and early 90's, can be viewed as one practical realization of this idea. In MUC data, newswire stories are annotated with one of several templates describing high-level events and their properties; for example, one MUC corpus consists of news stories about terrorist



John went to Lundy's. He ordered lobster.
He paid the check and left.

KS
John **$RESTAURANT**
Customer = John
Food = lobster
Name = Lundy's
Location = Brooklyn
CD
S1 John **PTRANS** John **to** restuarant
E
S1 John **MTRANS** waitress **ATRANS** lobster **to** John **to** waitress
rE
S1 John **INGEST** lobster **to** stomach
rE
S1 John **ATRANS** money **to** management
rE
S1 John **PTRANS** John from restaurant

Thus an entire story spanning many script and non-script-like events would be represented as a linked causal chain of Conceptual Dependency conceptualizations, some subset of which would be linked via the Script link to the scriptname that governs it at the Knowledge Structure level.

(a) Example from Schank and Abelson (1977)

The terrorists ***used*** *explosives against the town hall. El Comercio reported that alleged Shining Path members also* ***attacked*** *public facilities in huarpacha, Ambo, tomayquichua, and kichki. Municipal official Sergio Horna was seriously* ***wounded*** *in an explosion in Ambo.*

The entities from this document fill the following slots in a MUC-4 bombing template.

Perp: Shining Path members **Victim**: Sergio Horna
Target: public facilities **Instrument**: explosives

(b) Example template-filling for Message Understanding Conference (MUC) data; from Chambers and Jurafsky (2011).

Kidnap Template (MUC-4)

Perpetrator *Person/Org* who releases, abducts, kidnaps, ambushes, holds, forces, captures, is imprisoned, frees

Target *Person/Org* who is kidnapped, is released, is freed, escapes, disappears, travels, is harmed, is threatened

Police *Person/Org* who rules out, negotiates, condemns, is pressured, finds, arrests, combs

Weapons Smuggling Template (NEW)

Perpetrator *Person/Org* who smuggles, is seized from, is captured, is detained

Police *Person/Org* who raids, seizes, captures, confiscates, detains, investigates

Instrument A *physical object* that is smuggled, is seized, is confiscated, is transported

(c) Two example frames (a.k.a. templates) learned in Chambers and Jurafsky (2011). The names of frames and roles were given manually by the researchers, but their noun types, and the typical verbs they are arguments of, are learned automatically from a raw unlabeled corpus.

activities in Latin America, with a number of templates for the high-level events such as "bombing," "kidnapping," etc.; see Figure 1b for an example. One possible application is structured corpus summarization.

We pursue unsupervised learning to automatically induce frames from text, and to annotate a given text with the frames that underlie it. Frame-like structured representations have been proposed to model beliefs, opinions, worldviews, and many other aspects of knowledge, but we focus on co-occurring events and actor roles, following on the work of Chambers and Jurafsky (2011), which developed an unsupervised learning approach to induce MUC-style frames from newspaper articles.[1]

Their results are very interesting (Figure 1c): they were able to reconstruct some of the hand-built frames from the MUC dataset, and discovered several other ones as well. Their approach consists of an ad-hoc cascade of several stages of clustering algorithms and many heuristic steps; unfortunately, it is difficult to replicate, reason about, and extend.

We develop an alternative approach, within the framework of Bayesian generative statistical models, to accomplish this task. By taking this approach, we can access a variety of (fairly) well-understood statistical algorithms for learning and inference, and have principled methods to extend and modify the model to incorporate document metadata and other sources of information of great interest for text analysis. We begin by describing previously existing models for documents and syntactic tuples, then describe our approach, which builds on these previous advances.

3 Document-word mixture model: LDA

The well-known model of latent Dirichlet allocation (Blei et al., 2003) fits latent mixtures of words over a corpus of documents. Documents are represented as a mixture of high-level topics, while their words are generated by the topic-word distributions.

One possible hierarchical generative story for LDA is:

- Fix K number of topics, and priors α, β
- **Topic lexicon:** For each topic $k = 1..K$,
  - Draw $\phi_k \sim Dir(\beta)$
- **Document-word data:** For each document $d = 1..D$,
  - Draw $\theta_d \sim Dir(\alpha)$
  - For each word token $i = 1..N_d$,
    - Draw $z_i \sim \theta_d$
    - Draw $w_i \sim \phi_{z_i}$

where

- V = the number of distinct words types under consideration, K = the fixed number of topics, and D the number of documents in the corpus;

[1] Note that in both Chambers and Jurafsky (2011), and the MUC approach in general, there is no temporal or causal ordering to the fine-grained events in a single text; this contrasts with the usual notion of a script or narrative. In this work, we stick to this atemporal, "co-occurring events and actors" view of a "script," since it is simpler to model. Therefore we use the term "frame" in the rest of the document to avoid confusion (though that term is ambiguous in other previous literature).

- θ_d parameterizes a document's discrete distribution over topics; i.e. $\theta_d \in Simplex(K)$.
- ϕ_k parameterizes a topic's discrete distribution over words; i.e. $\phi_k \in Simplex(V)$.

An interesting way to think about LDA is to view the dataset as a giant list of (DocumentID, WordID) pairs for every token: (d_i, w_i) where i indexes across all tokens in the corpus.[2] The likelihood of all words (conditioning on the assignment of tokens to documents, which is taken as fixed) is:

$$\begin{aligned} p(w|d) &= \prod_i p(w_i|d_i) && (1) \\ &= \prod_i \sum_{z_i} p(w_i|z_i)\, p(z_i|d_i) && (2) \end{aligned}$$

An effective approach to inference in LDA is collapsed Gibbs sampling (Griffiths and Steyvers 2004, Neal 1992), which only explores settings of the per-token latent variables z, "collapsing out" the θ and ϕ multinomials by exploiting Dirichlet-multinomial conjugacy. The collapsed Gibbs sampler holds a setting for all z variables, and sequentially updates one z_i variable at a time, by resampling according to its probability conditioning on all other variables:

$$p(z_i \mid z_{-i}, w, d; \alpha, \beta) \;\propto\; p(z_i \mid z_{-i}, d, \alpha)\, p(w_i \mid z_i, w_{-i}, \beta) \tag{3}$$

where z_{-i} and w_{-i} denote all topic indicators and word tokens at all other positions besides i. This approach also conditions on the Dirichlet document-topic and topic-word priors α and β (but not θ or ϕ, which are implicitly integrated out).

Due to Dirichlet-multinomial conjugacy, the counts of documents, topics, and words are sufficient statistics to represent the two probabilities above as:[3]

$$p(z_i = k \mid z_{-i}, w, d; \alpha, \beta) \;\propto\; \frac{C[k, d_i] + \alpha}{C[d_i] + \alpha_0}\, \frac{C[w_i, k] + \beta}{C[k] + \beta_0} \tag{4}$$

where C represents a count over all token positions except for i: $C[k, d_i]$ denotes the count of how many tokens in the document have topic assignment k, $C[d]$ is how many tokens the document contains, $C[k]$ is how many tokens are assigned to topic k across the whole corpus, and $C[w_i, k]$ is how many tokens under topic k are the word w_i. These are just marginal counts of the token-level triples (d, z, w).

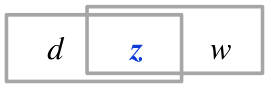
[And: $\alpha_0 = K\alpha, \beta_0 = V\beta$; they are the concentration parameters associated with these symmetric Dirichlet priors. For implementation, $(C[d_i] + \alpha_0)$ can be omitted since it is invariant in k.]

This gives a clear interpretation to the probability of assigning z_i to topic k: the left term favors topics that are already prevalent within the document, while the right term favors topics for which that word is common. (The constraints are probabilistic and mediated by Bayes rule.) They might be called document-topic and topic-lexical coherency constraints, since they tend to favor a specific (soft) set of topics for one document, and a specific (soft) set of words for one topic; and they correspond to the two factors in a factor graph view of the probabilistic model that we illustrate in Figure 1.

The collapsed Gibbs sampling algorithm explores the posterior by taking multiple iterations through the data, updating the z_i variables in turn, while maintaining the necessary marginal count tables. The factors in the Figure 1 diagram correspond to the highest-dimension count tables that are necessary to maintain. We will use this simple inference framework for all our models.

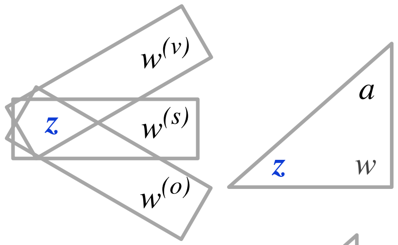
[2]Asuncion et al. (2009) and Hoffman (1999) take this view.

[3]Details in section 9.



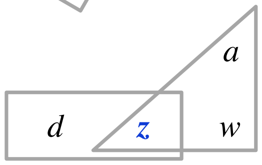
Document-Word LDA

$$p(\mathbf{z} \mid d, w) \;\propto\; p(\mathbf{z} \mid d)\; p(w \mid \mathbf{z})$$



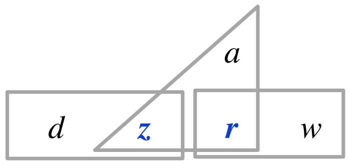
Tuple-only (Rooth-style) model

$$p(\mathbf{z} \mid w, a) \;\propto\; p(\mathbf{z}) \prod_a p(w^{(a)} \mid \mathbf{z}, a)$$



Document-Tuple (Model 1)

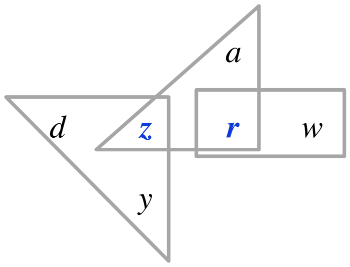
$$p(\mathbf{z} \mid d, w, a) \;\propto\; p(\mathbf{z} \mid d) \prod_a p(w^{(a)} \mid \mathbf{z}, a)$$



Document-Tuple-Role (Model 2)

$$p(\mathbf{z} \mid d, r, a) \;\propto\; p(\mathbf{z} \mid d) \prod_a p(r^{(a)} \mid \mathbf{z}, a)$$

$$p(\mathbf{r^{(a)}} \mid z, w^{(a)}, a) \;\propto\; p(\mathbf{r^{(a)}} \mid z, a)\; p(w^{(a)} \mid \mathbf{r^{(a)}})$$



Metadata extension (PLDA) (Model 3)

$$p(\mathbf{z} \mid d, y_d, r, a) \;\propto\; p(\mathbf{z} \mid d, y_d) \prod_a p(r^{(a)} \mid \mathbf{z}, a)$$

Figure 1: Factor diagrams for the described models, along with their respective the Gibbs update conditional factorizations for their latent variables. Latent variables are **blue**; the rest are observed. A factor graph is formally defined as a bipartite graph between variables and factors; a factor is visually represented here as a shape surrounding the variables it implicates. These diagrams only show the discrete variables in the models; their Dirichlet-multinomial priors are not shown.

4 Semantic tuple mixture models

Using LDA to model the bag-of-words in a document can recover impressive topical clusters of words, but our goal is to retrieve deeper frames involving systems of typical events and relationships between actors in a text. The key information is relational, and language encodes predicate-argument relationships in abundance—at the most basic level, as verbs and their arguments. Verbs often encode events or actions, while nouns usually encode entities. When a sentence expresses a tuple of (verb, subject, object), this often encodes an action (verb) performed by an agent (subject), upon or with the object. Teasing apart these relationships is complicated, and extracting more general forms of the roles from text is its own subproblem in natural language processing (semantic role labeling (TODO CITE)); but focusing on verbs, subjects, and objects is sufficient to get started in modeling script-like frames; indeed, this is the focus of the Chambers and Jurafsky work.

There exists a line of work on modeling verb-argument tuples, usually under the guise of learning *selectional restrictions*, which means the typical classes of arguments for a verb. For example, "The man bit the dog" is unnatural because the subject and object arguments of the verb *bite* tend to take different classes of noun arguments than are used here. This line of work always focuses on modeling (verb, noun) tuples individually—often extracted from a large corpus with a syntactic parse, as a preprocessing step—without considering document context.

Looking just at work that uses generative clustering approaches, Rooth et al (1999) and Pereira et al (1993) propose a simple class-based model where (verb, object) pair belongs to a particular class, and each class has separate verb and object word distributions. Rooth illustrates the model learns broad classes of verbs; for example, one learned class has its most probable verbs as *increase as, fall, pay, reduce, rise* while the the most probable objects are *number, rate, price, cost, level*; they interpret this class as "scalar change."

We can generalize the Rooth model to (verb, subject, object) triples, and add some Dirichlet priors, as follows. $\phi_k^{(arg)}$ denotes a word multinomial for argument type arg and class k; and there are three argument types (verb, subject, object), treated completely separately.

- **Frame lexicon:** For each $k = 1..K$, sample three word multinomials: $\phi_k^{(v)}, \phi_k^{(s)}, \phi_k^{(o)} \sim Dir(\beta)$
- **Tuple data:** For each tuple $i = 1..N$,
  - Draw its class indicator z_i (from a fixed prior),
  - Draw the three words from their repsective multinomials: $w_i^{(v)} \sim \phi_{z_i}^{(v)}$; $w_i^{(s)} \sim \phi_{z_i}^{(s)}$; $w_i^{(o)} \sim \phi_{z_i}^{(s)}$

A factor diagram is shown in Figure **??**. This approach models every tuple independently, with no document or other context; the only thing shared across tuples are the per-class argument word distributions. More recent work extends Rooth to model other syntactic relation pairs (O'Seaghdha 2010) and web-extracted triple relations (Ritter and Etzioni 2010). The Bayesian framework makes it easy to explore many variants; we illustrate several of these proposed models in Figure 2.

There is still more previous work, including Grenager and Manning (2006) and Titov and Klementiev (2011) which use a broadly similar Bayesian mixture modeling approaches to cluster syntactic relations and classes for verb arguments and other syntactic dependencies, arguing this recovers deeper semantic categories of verbs and other relationships. Lang and Lapata (2010) introduce a latent logistic regression model for argument structure.

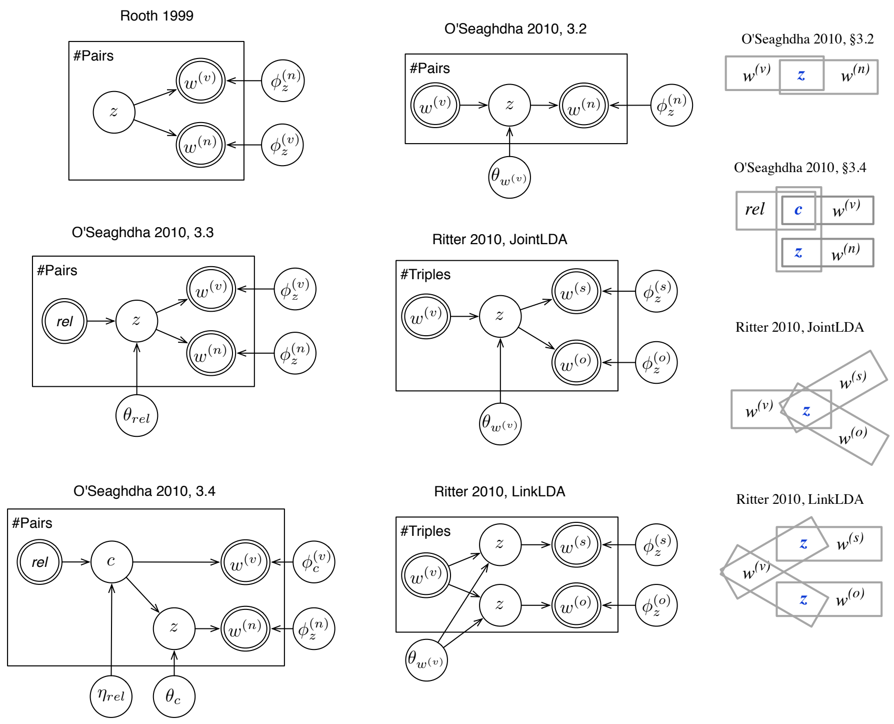


Figure 2: Several relational tuple models proposed in previous work. Double-border variables are observed. Multinomials are shown but Dirichlet priors omitted. The unified notation necessarily conflates distinctions between the authors; for example, Ritter models with web-extracted semantic tuples, not syntactic V-S-O tuples.

5 Mixture models of documents, tuples, and roles

We present a succession of three models to capture aspects of unsupervised learning of frames. Initially, we preprocess a corpus of documents into syntactic (verb, subject, object) tuples using a syntactic parser,[4] yielding a dataset of tuples $(d, w^{(v)}, w^{(s)}, w^{(o)})_{i=1..N}$. Our models are fit to this data.

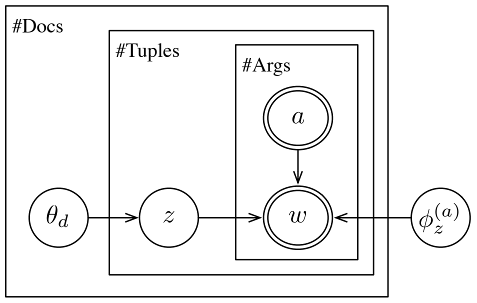
Figure 3 illustrates the models in probabilistic DAG notation (Koller and Friedman 2010, Bishop 2006); see also the factor graph diagrams in Figure 1.

5.1 Model 1: Document-Tuple

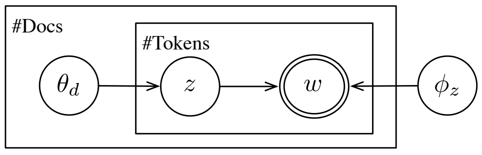
First, we extend the Rooth independent tuple model to incorporate document context, by giving tuples within the same document a shared multinomial prior over classes. This amounts to modifying LDA to replace its word generation process with a V-S-O tuple generation process.

For clarity, we notate the three verb, subject, and object syntactic argument positions with integers $1, 2, 3$, and within a tuple reference them with an observed variable $a \in \{1, 2, 3\}$. We also now call the latent variables "frames." The generative process is:

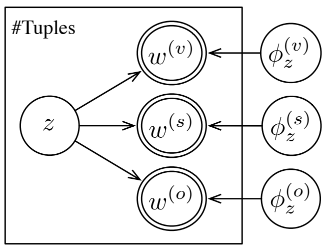
[4] Stanford Parser and Dependencies: http://nlp.stanford.edu/software/ (Klein and Manning 2003, de Marneffe et al 2006)



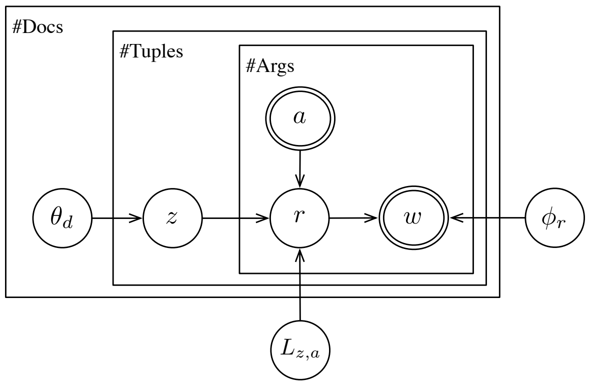
Model 1: Document-Tuple



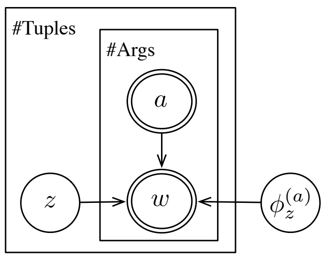
Document-Word LDA



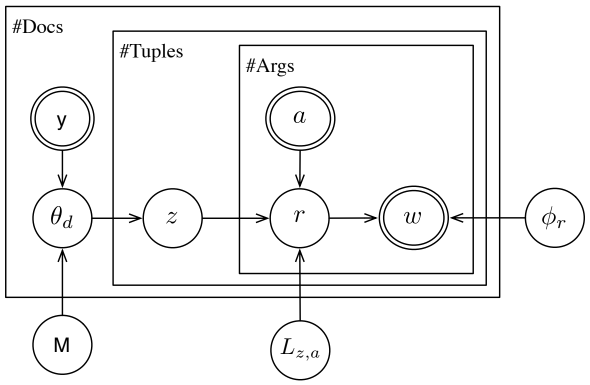
Tuple-only (Rooth-style) model



Model 2: Document-Tuple-Role



Alternate notation for the tuple-only model



Model 3: Metadata-frame constraints (PLDA)

Figure 3: Probabilistic DAG notation for our models and related models. Double-border variables are observed. Dirichlet priors α, β (and for Model 2, γ) are omitted, but all multinomials are shown. Compare to factor diagrams in Figure 1.

- Fix K frames, and Dirichlet priors α, β
- **Frame lexicon:** For each frame $k \in 1..K$, and argument position $a \in \{1,2,3\}$,
  - Draw word multinomial $\phi_k^{(a)} \sim Dir(\beta)$
- **Document-tuple data:** For each document $d \in 1..D$,
  - Draw frame multinomial $\theta_d \sim Dir(\alpha)$
  - For each tuple i in the document,
    - Draw frame indicator $z_i \sim \theta_d$
    - Draw word triple: for each argument position $a \in \{1,2,3\}$,
      - Draw $w_i^{(a)} \sim \phi_{z_i}^{(a)}$

The per-document prior has the effect of balancing between tuple coherency and document coherency. The Gibbs update step is to sample from the conditional distribution

$$p(z_i \mid w, z_{-i}, d, \alpha, \beta) \propto \; p(z_i|d,\alpha)\; p(w^{(v)}|z_i,\beta)p(w^{(s)}|z_i,\beta)p(w^{(o)}|z_i,\beta) \tag{5}$$

where, as in LDA,

$$p(z_i|d,\alpha) = \frac{C[z_i,d]+\alpha}{C[d]+\alpha_0}$$

(though the count is over tuples, not word tokens.)

As α increases, the $p(z_i|d)$ term is encouraged to be more and more uniform, and thus have less of an impact on $q(z_i)$; it collapses into the Rooth tuple independence model for $\alpha \to \infty$. For a moderate value of α, the model encourages tuples in the same document to share the same class. We hypothesize this will encourage the classes to be not just about linguistic argument categories (as previous tuple modeling work has attempted to model), but also capture related types of events and actions in the corpus, in the same way that LDA on individual words can capture topical themes.

5.2 Model 2: Document-Tuple-Role

In Model 1, every class or "frame" has completely independent word distributions for the argument positions. But ideally, we would like higher level roles that summarize word distributions but cut across the particular types of events. For example, in the domain of crime stories, we might want to learn roles for POLICE, CRIMINAL, and CRIME and know that typical events might include [v=ARREST, s=POLICE, o=CRIMINAL] and [v=COMMIT, s=CRIMINAL, o=CRIME]; those roles might be typically instantiated by words such as *police, cops, authorities* for POLICE, or *perpetrator, attacker, thief* for CRIMINAL, etc.

We seek to accomplish this by introducing latent *role* variables. Every role associates with a word multinomial — a "role-filler" distribution — while each frame tends to see certain roles in different argument positions (the "linker" distribution). The roles intermediate between frames and words, allowing cross-cutting roles across frames.

The generative process for Model 2 is:

- Fix R roles, K frames, and Dirichlet priors α, β, γ
- **Frame lexicon:**

- For each role $r \in 1..R$,
  - Draw word multinomial $\phi_r \sim Dir(\beta)$
- For each frame $k \in 1..K$, and argument position $a \in \{1, 2, 3\}$,
  - Draw the "linker" $L_{k,a} \sim Dir(\gamma)$, a multinomial over roles: $L_{k,a} \in Simplex(R)$.

- **Document-tuple data:**
  - For each document $d \in 1..D$, draw frame multinomial $\theta_d \sim Dir(\alpha)$
  - For each tuple i for document d,
    - Draw frame indicator $z_i \sim \theta_d$
    - Draw word triple: for each argument $a \in \{1, 2, 3\}$,
      - (if a is not present in this tuple, skip)
      - Draw role $r^{(a)} \in L_{z_i,a}$
      - Draw word $w^{(a)} \in \phi_{r^{(a)}}$

(We have slightly different handling of NONE values than in Model 1: they are skipped and not modeled at all. If they are explicitly modeled, they take a very large proportion of the probability mass in the role-word distributions, making the results harder to interpret.)

We use a Gibbs sampler that computes updates separately for frame and role variables:

$$p(z_i \mid d, z_{-i}, r, a; \alpha, \gamma) \propto p(z_i \mid d, z_{-i}, \alpha) \prod_a p(r^{(a)} \mid z_i, a, \gamma) \tag{6}$$

$$= \frac{\#T[z_i, d] + \alpha}{\#T[d] + \alpha_0} \prod_a \frac{\#W[r^{(a)}, z_i, a] + \gamma}{\#W[z_i, a] + \gamma_0} \tag{7}$$

$$p(r^{(a)} \mid z, w^{(a)}, a; \gamma, \beta) \propto p(r^{(a)} \mid z, a, \gamma) p(w^{(a)} \mid r^{(a)}, \beta) \tag{8}$$

$$= \frac{\#W[r^{(a)}, z_i, a] + \gamma}{\#W[z_i, a] + \gamma_0} \frac{\#W[w^{(a)}, r^{(a)}] + \beta}{\#W[r^{(a)}] + \beta_0} \tag{9}$$

where #T denotes a count over tuples (excluding tuple i), and #W denotes a count over word tokens; in the first equation, it excludes counts from the three words at tuple i; in the second equation, it excludes the word at position a in tuple i.

The algorithm we use is, for one pass through the data: For each tuple i, (1) update z_i, then (2) for $a = 1..3$, update $r^{(a)}$.

Mixing is substantially slower for Model 2 than Model 1, probably due to the strong dependency between the frame and word roles at a single tuple. It may be important in the future to investigate alternative inference procedures that can make larger moves, such as the split-merge sampler (Jain and Neal 2004).

[BTO: TODO figure out if/how Model 2 is related to the chinese restaurant franchise view of the HDP]

5.3 Model 3: Metadata-Frame associations

Many corpora of analytic interest have some form of observed *document labels*, such as the document's author, year of publication, gender of the author, political slant of the publication, etc. Corpus analysis usually involves associating document labels with text (e.g. topical trends over time, difference of writing styles between genders, etc.) The framework of Partially Labeled LDA

(Ramage et al., 2011) learns to associate latent topics with these observed data through a very simple mechanism of constraining certain topics to only appear in documents having certain labels. We can easily extend it to our frame models.

PLDA assumes there are J possible binary labels; any set of them may be on for a particular document, described by binary vector $y^{(d)} \in \{0,1\}^J$. Every label is globally fixed to be associated with a particular subset of topics. A document's topics are constrained to only include ones that are allowed by the label-topic associations.

Let $M \in \{0,1\}^{J \times K}$ be a binary matrix describing the label-topic associations, such that $M_{jk} = 1$ means that a document with label j is allowed to use topic k. For example, say $J = 3, K = 9$ and

$$M = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

indicating that y_1 associates with the first 5 topics, while y_2 and y_3 associate with two topics each. Say that y_1 is a "background" label always on for all documents, while y_2 and y_3 indicate whether the author is male or female, respectively. Then a male author can draw from topics $\{1..7\}$, while a female author can draw from topics $\{1..5\} \cup \{8, 9\}$.

The only change to document generation is to sample the document-topic multinomial $\theta^{(d)} \in \mathbb{R}^K$ from a masked prior vector $\alpha^{(d)} \in \mathbb{R}^K$, which is zeroed out where the document labels forbid it. For document d, having an observed binary label vector $y^{(d)}$,

- Let $\alpha^{(d)}$ be such that $\alpha_k^{(d)} = \alpha$ if $M_{jk} = 1$ for any j having $y_j^{(d)} = 1$; otherwise $\alpha_k^{(d)} = 0$.
- Draw $\theta^{(d)} \sim Dir(\alpha^{(d)})$

If the rows of M are mutually exclusive—i.e. every label's topic set doesn't overlap with any others—the masking operation is equivalent to a matrix multiply $\alpha^{(d)} = (y^{(d)})^\intercal M$. (This suggests a connection between PLDA and the Dirichlet-multinomial regression topic model (Mimno and McCallum, 2008).)

The collapsed Gibbs sampler is trivially easy to implement for PLDA since conjugacy is preserved; one simply zeroes out the disallowed topics when building the conditional Gibbs sampling distribution for z_i. This makes it easy to add PLDA to either Model 1 or Model 2, to associate frames with document characteristics.

A weakness of PLDA is that the label-topic architecture, represented in M, has to be specified in advance. This can be complicated to tune. Ramage et al. used a setup of several background topics, and a fixed number of topics each for each label, with each labels' topics exclusive. An interesting extension for future work is to use a Dirichlet process prior to automatically choose the number of topics each label is associated with (also examined by Ramage et al.).

5.4 Posterior inference for hyperparameters

For exposition, we have indicated the Dirichlet priors α, β, γ are fixed. Specifying them manually is, to a certain extent, useful, since they control the degree of sparsity observed in the posteriors—for example, a small α will tend to decrease the number of topics used in a document. But it is possible to fit them to the data as well, to either maximize likelihood (Wallach 2009, Asuncion 2009, Minka 2009) or compute a posterior over them (Johnson 2008, TODO find others); these authors have found that either of these approaches tends to improve model fits.

During inference, we sample these Dirichlet concentration parameters through slice sampling (Neal 2003),[5] an MCMC method for continuous variables that, like Metropolis-Hastings, allows one to sample from a complicated distribution that can only be evaluated at individual points. But unlike Metropolis-Hastings, it requires no proposal distribution, and very little width tuning, as it adaptively learns a step size.

We interleave the core z and r Gibbs update iterations with hyperparameter Gibbs update steps, using the unnormalied posteriors. For Models 1 and 2, the α update is through:

$$p(\alpha \mid z) = p(\alpha) \prod_d p(z_{\{i:\, d_i = d\}} \mid \alpha)$$

where $p(\alpha)$ is assigned to be a vague prior, and the document likelihood term $p(z_{\{i:\, d_i=d\}} \mid \alpha)$ is the "single-path" Dirichlet-compound multinomial, derived in Section 9. Update steps for β and γ are analogous. For the prior $p(\alpha)$, many authors use a vague Gamma prior that is relatively flat across all positive reals; but we found success with the even simpler approach of having no prior and only using the likelihood term as the unnormalized posterior; this corresponds to an improper uniform prior having constant density on $\mathbb{R}$. See section 6.5 for details and discussion.

6 Experiments

6.1 Dataset

We have implemented models 1 and 2 and experimented fitting them on the news articles from the LDC New York Times Annotated Corpus (Sandhaus, 2008). This corpus is useful because news text is relatively easy for current syntactic parsers to analyze, and the articles have been extensively categorized by hand—every article has a number of descriptor labels. This makes it easier to extract subsets that are targeted towards specific topics, making the models' output easier to interpret. Also, the labels can be used to experiment with Model 3.

Inspired by examples of narrative event structures in the domain of crime reporting (Chambers and Jurafsky 2009), and examples in earlier work on scripts (DeJong 1982), we select news articles for which any of their category labels contain any of the words *crime*, *crimes*, or *criminal*. This yields 53,182 articles. We show the most common matching labels, and a sampling of headlines, in Table 2.

We have been using this dataset, and a smaller development subset of it (2,602 articles) for initial experiments. Unfortunately, upon further review, there seem to be a number of data quality issues with the metadata—in particular, there are two different system of categories, from different groups within the New York Times, and one of them seems to have generally higher quality labels; for example, they disagree whether "Crime and Criminals" applies to international terrorism or not, or stories about 9/11 memorials, etc. When extracting this particualr dataset, we collapsed together all categories, but for future experiments with the corpus, this and other metadata quality issues may require further consideration.

6.2 Parse preprocessing

All linguistic analysis is performed as a preprocessing step, by running the Stanford CoreNLP system,[6] which performs sentence segmentation, part-of-speech and named entity tagging, syntactic

[5] Using an adaptation of Iain Murray's implementation: http://homepages.inf.ed.ac.uk/imurray2/teaching/09mlss/slice_sample.m

[6] http://nlp.stanford.edu/software/corenlp.shtml

count	category label
48,645	crime and criminals
9,497	sex crimes
6,304	sentences (criminal)
3,892	war crimes, genocide and crimes against humanity
2,818	organized crime

Table 1: Most common category labels matching query

1987-05-05	JURY SELECTION MAJOR HURDLE IN TRIAL THAT MAY LAST YEARS
1988-02-25	Moslem Patrol Helps Cut Crime in Brooklyn
1991-10-22	GUILTY PLEAS SET IN U.S. COAL CASE
2001-05-30	4 GUILTY IN TERROR BOMBINGS OF 2 U.S. EMBASSIES IN AFRICA; JURY TO WEIGH 2 EXECUTIONS
2001-10-17	A Rush for Cipro, and the Global Ripples
2003-08-09	World Briefing — Europe: Northern Ireland: Fund For Bomb Lawsuits
2003-10-03	Bryant's Accuser Won't Have to Testify
2004-07-18	Despite Appeals, Chaos Still Stalks the Sudanese
2005-04-01	World Briefing — Europe: France: Longer Prison Term In Graft Case

Table 2: Sample of headlines from the dataset.

parsing (Klein and Manning 2003) and conversion to dependencies (de Marneffe et al 2006). A dependency parse explicitly represents syntactic relations between words, including the subject and object relationships between a verb and its arguments. We extract (verb, subject, object) tuples from wherever there is a verb that has at least one subject or object; if either is missing (e.g. an intransitive verb, or implied arguments), the word is assigned a placeholder "NONE" symbol. We use the Stanford Dependencies' nominative subject and direct object relations for this (*nsubj* and *dobj*).

We selected a random sample of extracted V-S-O tuples to manually assess for accuracy. A subset are shown in Table 3. We annotated 40 tuples, in context in their respective sentences, consulting the Stanford Dependency papers (de Marneffe et al 2006, 2008), which have clear linguistic definitions of the grammatical relations, their conventions for analyzing compound verbs, etc. Out of 40 tuples, we found 30 had the subject and/or object arguments correct; 6 had one or both wrong; and 4 were incomplete (missing either subject or object)—75% precision. We were somewhat surprised to see that most tuples are incomplete. Complete V-S-O tuples make up only 18.8% of the tuples, while 43% are (V,S,*NONE*) and 38% are (V,*NONE*,O).

Finally, we use a vocabulary cutoff to aid efficiency, limiting to the 10,000 most common words across all V-S-O positions. Words that fall below the threshold are replaced with an "OOV" symbol. We use the lemmatized (stemmed) forms of words, as analyzed by the parser. Unlike standard topic models, we do not need to remove stopwords, since this is built-in to the grammatical analysis: the verb-subject and verb-object relations exist directly between content words.

We ran CoreNLP on the 53,182 input articles; 52,254 were processed without error or the software crashing, yielding 2.6 million V-S-O tuples.

6.3 Model fitting

For both Model 1 and 2, we initialize latent variables to random values, then run the Gibbs sampler as described in Section 5. Two Gibbs sampling runs are shown for Model 1, on the small and large

correct?	text and V-S-O tuple
RIGHT: for "workers," only use single head word of the noun phrase	In less than an hour , the police and rescue unit (workers $)_{subj}$ [found $]_{verb}$ the (organ $)_{obj}$ in the tall grass of the field , packed it in ice and took it to the hospital .
RIGHT	Mrs. (Bissell $)_{subj}$, she said , never [learned $]_{verb}$ what her husband did with the money .
INCOMPLETE: "he" should be subject of "defrauded" since SD uses content verb as head	Asked why he had [defrauded $]_{verb}$ the insurance (company $)_{obj}$ and been partners with the mob . . .
WRONG: lists are a known problem case for current parsers	Guilty on Five Charges Mr. Garcia was found guilty on all five charges against him : (theft $)_{subj}$ of heroin , possession with intent to [distribute $]_{verb}$ heroin , narcotics conspiracy and two (counts $)_{obj}$ of money laundering .

Table 3: Example extracted tuples and our annotations in their original sentences (in tokenized form).

datasets with $K = 100$ frames in both cases. α is resampled, but β is fixed for efficiency reasons. (It is fixed to $\beta = 100$, approximately a value resulting from preliminary experiments when it was resampled.) We show trace plots for the sampling runs in Figure 7. It is notoriously difficult to assess MCMC convergence but we attempt some analysis.

Log-likelihood should increase as the Gibbs sampler moves towards a high-density posterior region; ideally, it should then mix, exploring the posterior. For the small dataset, it is possible to run 5000 iterations within several hours,[7] but unfortunately, it does not seem that log-likelihood has yet leveled off (especially when we view just the last 200 iterations). (On an even smaller dataset, also shown, it may have leveled off after 5000 iterations.) On the full dataset, 1000 iterations takes even longer (about 10 hours), and is quite far from convergence.

It should be noted that, even if the log-likelihood levels off, that does not guarantee the MCMC chain is actually mixing—it could be stuck in a local mode. To properly assess convergence, it is recommended to trace the movement of particular parameters over time. Unfortunately, in a mixture model the components have identifiability issues, since swapping their labels yields an equivalent likelihood solution; this makes it difficult to interpret movements in individual topic-word or document-topic distributions difficult.

This is disappointing; what is really going on is the Gibbs sampler is acting as an optimizer; if we stop it early, it is simply an incomplete optimizer. There is a strange phenonenon in the literature that uses Gibbs sampling for topic models: the algorithm is justified on theoretical MCMC grounds of exploring the posterior distribution; but in practice, since the dimensionality is so high, everyone uses it as a stochastic optimizing "mode-finder" that then infers a small piece of the posterior distribution around that mode. Our use of Gibbs sampling largely follows these lines, and suggests that variational or other approaches to inference may be useful.

Final note: as a supplement to log-likelihood, we experimented with tracking the number of changes to z variables during each iteration—a very simple measure of how much the Gibbs sampler is moving, which we might expect to be higher when it is in a low density region and "trying" to get out. This simple count is computationally attractive because it is substantially

[7]Implementation in C/Python.

faster to compute than the likelihood; we compute it every iteration. In early iterations, it rapidly decreases as likelihood is rapidly increasing, though by later iterations it is less clear. Over the iteations where we measure log-likelihood, its correlation with the number of changes is -0.94 (and about the same for log(number of changes)).

6.4 Effects of changing Dirichlet concentration paramters

TODO explain (1) 99% mass statistics for Dirichlet sparsity analysis, (2) simulation results, (3) empirical results on changing alpha on Model 1. Explain (concentration,mean) view of Dirichlets. Evidence that we are seeing the asymptotic convergence to Dirichlet process under fixed concentration paramter. (Is it Wallach 2009 that mentions this as well; high K, low α approximation of DP?) Explain (derive?) relationship of my "top-99% mass" statistic to asymptotic CRP theory of $O(\alpha\, logN)$ non-empty tables.

With the more complicated models, having several concentration paraemters, it becomes too difficult to tune all of them. This motivated the implementation of posterior sampling of the concentration parameters.

6.5 Slice sampling Dirichlet concentration parameters

This approach is described in section 5.4. We experimented with several approaches to scheduling the hyperparameter updates, since they are fairly expensive. In order to best increase overall likelihood over time, it seemed most useful to a "token burn-in" period at the start, leaving the hyperparameters at fixed values then having about 100 iterations of z and r Gibbs sampling iterations; this allows the model to learn reasonable frame distributions, which is the highest dimensional part of the model. Then we perform hyperparameter resampling every 50 iterations. It turns out this strategy is similar to the one implemented in the MALLET toolkit.[8]

For each hyperparameter, we run the slice sampler for 10 iterations, then take the last value of the parameter as a sample from its posterior. (Johnson 2008 uses 20, and Mallet uses [BTO: TODO check] by default.) To choose the number 10, we performed a simulation experiment, generating 500 artificial Dirichlet-multinomial datasets, and ran a slice sampler on each, assessing convergence of the distribution across chains to the true distribution. Figure 8 shows, for each iteration, the distribution of the 500 chains' states against the true posterior, as calculated with a grid approximation on the DCM likelihood. (This experiment was inspired by Cook et al. (2006)'s simulation testing technique for posterior inference.) [BTO: TODO redo and report parameter details. or better yet, run it on a real z sample]

The hyperparameter sampling substantially improves likelihood, as noted in previous work. Interestingly, most of the movement tends to happen early in the MCMC chain, then the hyperparameter stabilizes as the rest of the model is still moving. We checked if the outcome was initializer dependent by starting three different MCMC chains that were identical except for three different α initializers. Reassuringly, they all converged on the same region of values. This robustness to initialization was exactly what we wanted.

verbs	subjects	objects
serve, receive, give, impose, face, appeal, spend, seek, get, reduce	NONE(0.69), judge, OOV, sentence, defendant, court, prosecutor, man, jury, offender	sentence, term, year, NONE(0.03), time, penalty, probation, conviction, month, fine
OOV, rule, agree, argue, have, require, apply, refuse, hold, come	Court, court, judge, OOV, lawyer, law, case, decision, state, prosecutor	NONE(0.94), OOV, judge, request, time, case, court, hearing, defendant, decision
sell, use, buy, possess, distribute, fight, take, deal, get, carry	NONE(0.85), dealer, police, mother, percent, teen-ager, friend, OOV, people, Barry	drug, cocaine, crack, OOV, heroin, car, worth, marijuana, gas, food
make, approve, announce, increase, finance, have, propose, support, expand, raise	NONE(0.73), city, official, plan, Dinkins, Council, Mayor, Koch, administration, budget	plan, program, proposal, tax, force, budget, system, OOV, increase, NONE(0.01)

Table 4: Example output from Model 1. Four of 234 frames are shown; the ten most probable words are shown for each slot type. The top ten words take up 30% to 95% of the probability mass of the distribution, for these examples.

7 Results

7.1 Frame lexicons

Several frames from Model 1 are shown in Table 5. Several interesting classes of actions are extracted.

[BTO: TODO: analyze indpendent-tuple versus tuple-in-document. ($\alpha = \infty$ versus slice-sampled α). Is it more linguistic? Also, is there a good way to fairly compare topics between models? greedy 1-1 matching then qualitatively analyze the top-5 pairs?]

Model 2 extracts both role-word distributions, and frames with their frame-argument-role distributions. Unlike Model 1, it has two tuning parameters: K, the number of frames, and R, the number of roles. Inference is also slower. After experimenting with various settings, we found that $K = 10, R = 100$ gave reasonable results: the number of roles was large enough to give interesting word clusters, but unfortunately, the frames are less intelligible. We would like to have a large number for both K and R, but unfortunately, inference is even slower; after 30 hours of runtime $K = 10, R = 100$ seem to only be starting to mix (Figure **??**), but $K = 100, R = 100$ is much worse.

We show a Model 2 posterior in Figure 11. The role-frame matrix shows the counts that implicitly parameterize the linker $L_{z,a}$ multinomials: the three-dimensional count table $C[frame, arg, role]$. One row is a role. The top words in its multinomial are shown. Then for each frame, there are counts over the different argument positions it can appear in. The model naturally segregates verb-only roles from noun roles (since it is relatively rare for the same lemmatized word to be both a noun and a verb). Furthermore, it learns a sparse set of frame-role linking multinomials—the 0's in this matrix. The sparsity is controlled by the linker's Dirichlet parameter $gamma$, which is being resampled here. We show an interesting noun role, that appears in multiple frames, and

[8] http://mallet.cs.umass.edu/

verbs	subjects	objects
serve, receive, give, impose, face, appeal, spend, seek, get, reduce	NONE(0.69), judge, OOV, sentence, defendant, court, prosecutor, man, jury, offender	sentence, term, year, NONE(0.03), time, penalty, probation, conviction, month, fine
OOV, rule, agree, argue, have, require, apply, refuse, hold, come	Court, court, judge, OOV, lawyer, law, case, decision, state, prosecutor	NONE(0.94), OOV, judge, request, time, case, court, hearing, defendant, decision
sell, use, buy, possess, distribute, fight, take, deal, get, carry	NONE(0.85), dealer, police, mother, percent, teen-ager, friend, OOV, people, Barry	drug, cocaine, crack, OOV, heroin, car, worth, marijuana, gas, food
make, approve, announce, increase, finance, have, propose, support, expand, raise	NONE(0.73), city, official, plan, Dinkins, Council, Mayor, Koch, administration, budget	plan, program, proposal, tax, force, budget, system, OOV, increase, NONE(0.01)

Table 5: Example output from Model 1. Four of 234 frames are shown; the ten most probable words are shown for each slot type. The top ten words take up 30% to 95% of the probability mass of the distribution, for these examples.

in different ways—in frame 0 it is usually the subject, but in frames 2 and 3 it is more often the object.

[BTO: Still want to try PLDA if there's time, but is lower priority than getting the role model to work.]

7.2 Evaluation

[BTO: TODO: try to evaluate against PropBank, MUC, and/or VerbNet/FrameNet.]

[BTO: Idea: we hypothesize that the lexicons based in linguistic theory (PropBank, VerbNet, FrameNet) differ from information extraction datasets (MUC, ACE, Freitag Acquisitions...) in terms of domain specificity. We propose this can be tested through the α concentration parameter: if document context doesn't matter and we collpase into a Rooth-like sea-of-tuples model, everything becomes purely syntactic linguistics, so the model should be more like Propbank. But with stronger document-level constraints, the model becomes more like a topic model, and can learn specifics about the domain. Maybe.]

8 Conclusion

[BTO: Obvious TODO: write a conclusion]

9 Appendix: Dirichlet-multinomial conjugacy and the DCM

Consider the two-stage model

$$\theta \sim Dir(\alpha), \;\; x \sim Multinom(\theta)$$

where α is a real-valued vector Dirichlet parameter, and x is a vector of outcome counts. Let $A = \sum_k \alpha_k$; this is the concentration parameter.

$p(x|\alpha) = \int p(x|\theta)p(\theta|\alpha)d\theta$ is the Dirichlet Compound Multinomial, a.k.a. Multivariate Polya distribution. It is a distribution over count vectors, just like the multinomial, except it has the capacity to prefer different levels of sparseness vs. non-sparseness.

First note the Dirichlet density

$$p(\theta|\alpha) = \frac{\Gamma A}{\prod \Gamma \alpha_k} \prod \theta_k^{\alpha_k - 1} \quad = \quad \frac{1}{B(\alpha)} \prod \theta_k^{\alpha_k - 1} \tag{10}$$

where $B(\alpha)$ is the multivariate Beta function

$$B(\alpha) = \int \prod \theta_k^{\alpha_k - 1} d\theta = \frac{\prod \Gamma \alpha_k}{\Gamma A}$$

Now derive the DCM PMF:

$$\begin{aligned}
p(x|\alpha) &= \int p(x|\theta)\ p(\theta|\alpha)\ d\theta && (11)\\
&= \int Multinom(x;\theta)\ Dir(\theta;\alpha)\ d\theta && (12)\\
&= \int \left(\frac{N!}{\prod x_k!} \prod \theta_k^{x_k}\right) \left(\frac{1}{B(\alpha)} \prod \theta_k^{\alpha_k - 1}\right) d\theta && (13)
\end{aligned}$$

Because of conjugacy (i.e., the densities play nicely under multiplication), we can combine them into the integral of a new unnormalized Dirichlet, and rewrite into closed form.

$$\begin{aligned}
p(x|\alpha) &= \frac{N!}{\prod x_k!} \frac{1}{B(\alpha)} \int \prod \theta_k^{x_k + \alpha_k - 1} d\theta && (14)\\
&= \frac{N!}{\prod x_k!} \frac{B(\alpha + x)}{B(\alpha)} && (15)\\
DCM(x;\alpha) &\equiv \frac{N!}{\prod x_k!} \frac{\Gamma(A)}{\Gamma(A+N)} \prod \frac{\Gamma(\alpha_k + x_k)}{\Gamma(\alpha_k)} && (16)\\
&\text{(n. seq) (prob of a seq having counts } \vec{x}\text{)} && (17)
\end{aligned}$$

where $A = \sum \alpha_k$ and $N = \sum x_k$. To calculate this, one would rewrite the "number of sequences term" using $N! = \Gamma(N+1)$ and $x_k! = \Gamma(x_k + 1)$ then use the log-gamma function for everything. To calculate the log-probability of only a single sequence, omit the initial term $N!/\prod x_k!$; we call this a "single path DCM" or "DCM1":

$$DCM1(x;\alpha) = \frac{\Gamma(A)}{\Gamma(A+N)} \prod \frac{\Gamma(\alpha_k + x_k)}{\Gamma(\alpha_k)} \tag{18}$$

Note the DCM gives, for small α priors, a bowed-out preference to count vectors that lie on the extremes of the N-simplex—just like the Dirichlet with small α prefers bowed-out points on the 1-simplex. This can be seen as a preference for "bursty" behavior. A multinomial cannot do this: you only can specify the mean, then the variance is fixed. In the DCM, you control both mean and variance ($\approx$ inverse concentration), allowing burstiness a.k.a. over/under-dispersion.

For a single DCM draw where $N = 1$, instead of representing the draw as a sparse count vector x, we can represent it instead as an integer ID, $z \in \{1..K\}$. In the PMF, all the combinatorics drop away, leaving:

$$DCM(z; \alpha) = \frac{\alpha_z}{A}$$

which is simply the Dirichlet mean parameter. When there's only one draw there's no such thing as burstiness or not.

It is also easy to derive the posterior predictive distribution for a Dirichlet-Multinomial hierarchical model. We want to know the probability of the next draw z given the previous draws z_{-i}, using our entire posterior beliefs about θ. Represent z_{-i} as count vector $\vec{n} = (n_1..n_K)$:

$$p(z|z_{-i}, \vec{\alpha}) = \int p(z|\theta)\ p(\theta|z_{-i}, \alpha)\ d\theta \tag{19}$$

$$= \int Multinom(z; \theta)\ Dir(\theta; \vec{\alpha} + \vec{n})\ d\theta \tag{20}$$

The second step used Dirichlet-multinomial conjugacy. Now this is just the 1-draw DCM (i.e. the mean of the conjugately-updated Dirichlet),

$$p(z|z_{-i}, \vec{\alpha}) = DCM(z; \vec{\alpha} + \vec{n}) \tag{21}$$

$$= \frac{\alpha_z + n_z}{A + N} \tag{22}$$

9.1 DCM PMF in LDA hyperparameter sampling

Going through the full DCM is not necessary to derive the collapsed Gibbs sampling equations, but it *is* necessary for hyperparameter sampling, which requires evaluating the likelihood of the entire dataset under different hyperparamters α. LDA under collapsing can be viewed as a series of DCM draws:

- For each d, sample vector $z_{\{i:\ d_i=d\}} \sim DCMPath(\alpha)$
- For each k, sample vector $w_{\{i:\ z_i=k\}} \sim DCMPath(\beta)$

where "DCMPath" indicates choosing one random sequence having the counts of one DCM draw. (This could be computed by proceeding through a Polya urn process a.k.a. (finite) Chinese restaurant process.) Therefore, for their Gibbs update steps, the hyperparameter likelihoods are:

$$p(z \mid \alpha) = \prod_d DCM1(z_{\{i:\ d_i=d\}}; \alpha) \tag{23}$$

$$p(w \mid z, \beta) = \prod_k DCM1(w_{\{i:\ z_i=k\}}; \beta) \tag{24}$$

For the other models, analogous formulations are available as well. We were initially tempted to try to compute the likehoods with per-token local conditionals similar to what is used for the z Gibbs updates,

$$\prod_i p(w_i|z_i, \beta)\ p(z_i|z_{-i}; \alpha)$$

which is easy to compute, but unfortunately wrong: it is actually a pseudolikelihood approximation to the likelihood (Besag 1975). Since it is possible to compute the actual likelihood closed-form log-gammas, we do so.

However, it does turn out the running-sum pseudolikelihood is a good approximation, as shown in Figure 13, for correlating to likelihood across MCMC samples.

References

David Bamman, Brendan O'Connor, and Noah A. Smith. Censorship and content deletion in chinese social media, 2011. In submission to WWW-2012.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:9931022, 2003.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1277—1287, 2010.

David Lazer, Alex Pentland, Lada Adamic, Sinan Aral, Albert-Lszl Barabsi, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, Myron Gutmann, Tony Jebara, Gary King, Michael Macy, Deb Roy, and Marshall Van Alstyne. Computational social science. *Science*, 323(5915):721 –723, February 2009. doi: 10.1126/science.1167742. URL http://www.sciencemag.org/content/323/5915/721.short.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A Smith. From tweets to polls: Linking text sentiment to public opinion time series. In *International AAAI Conference on Weblogs and Social Media, Washington, DC*, 2010a.

Brendan O'Connor, Jacob Eisenstein, Eric P. Xing, and Noah A. Smith. A mixture model of demographic lexical variation. In *Proceedings of NIPS Workshop on Machine Learning in Computational Social Science*, 2010b.

Brendan O'Connor, David Bamman, and Noah A. Smith. Computational text analysis for social science: Model assumptions and complexity. In *Proceedings of the Second Workshop on Comptuational Social Science and the Wisdom of Crowds (NIPS 2011)*, 2011.

Daniel Ramage, Christopher D. Manning, and Susan Dumais. Partially labeled topic models for interpretable text mining. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, page 457465, 2011.

Evan Sandhaus. The New York Times Annotated Corpus, 2008.

Dani Yogatama, Michael Heilman, Brendan O'Connor, Chris Dyer, Bryan R. Routledge, and Noah A. Smith. Predicting a scientific community's response to an article. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 2011.

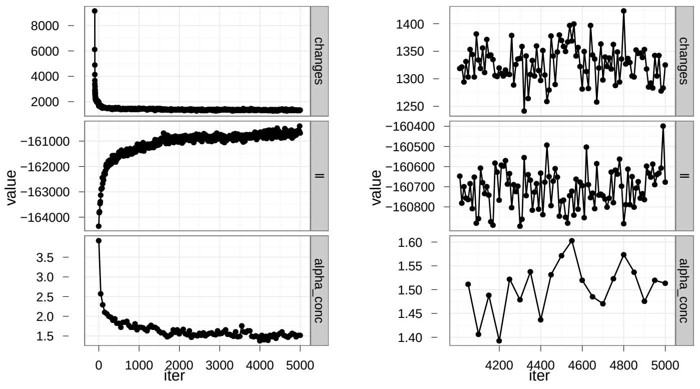


Figure 4: Tiny dataset (265 documents)

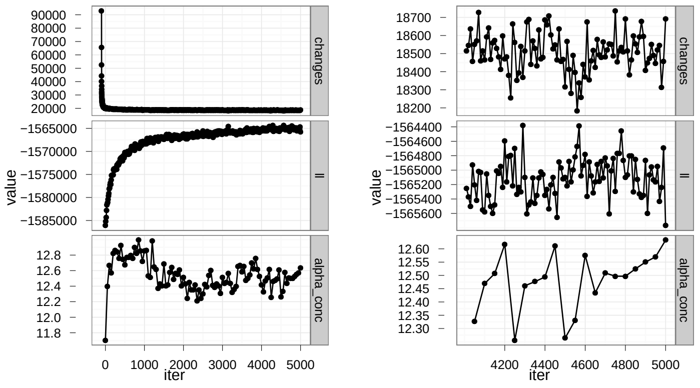


Figure 5: Small dataset (2600 documents)

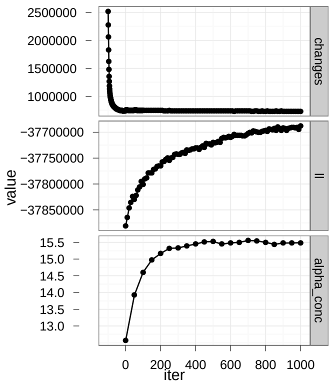


Figure 6: Full dataset (52000 documents)

Figure 7: Model 1 Gibbs sampling runs, on different sized datasets. The smaller datasets have 5000 iterations, and the large has only 1000. We show just the last 1000 iterations on the right. Statistics shown are are (1) number of z changes per iteration, (2) total $p(z, \alpha \mid w, \beta)$ log-posterior (calculated every 10 iterations), and (3) the α parameter, resampled every 50 iterations.

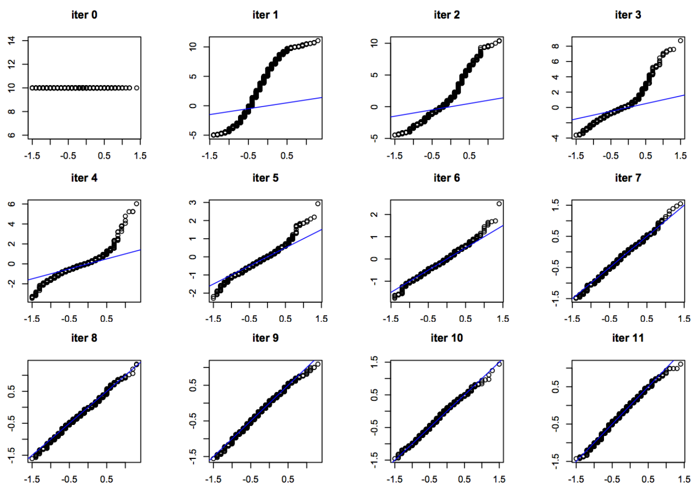
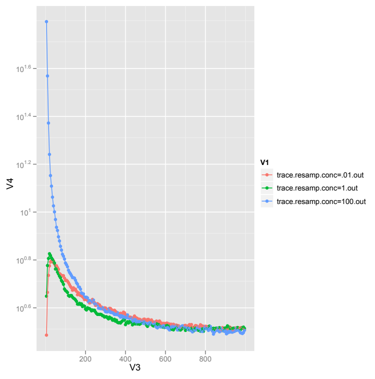


Figure 8: QQ-plots of the 500 simulated chains, showing their distribution converging to the true posterior over each slice sampling iteration. Each plot is the QQ-plot of 500 chain states, all at the same iteration.



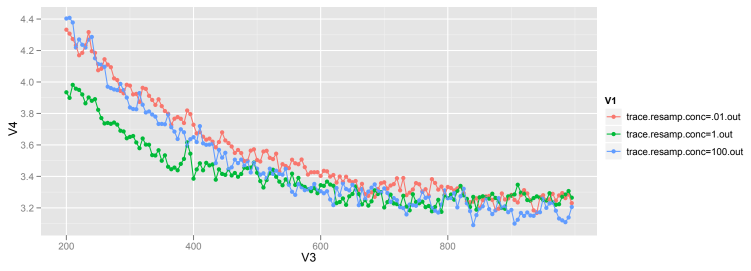


Figure 9: α parameter values over time, when being resampled, from three very different initial positions $\alpha = 0.01,\ 1,\ 100$. The left plot shows the entire history; the right shows from iteration 200 until the end. This is Model 1 on the small dataset. [BTO: TODO many obvious graph cleanups]

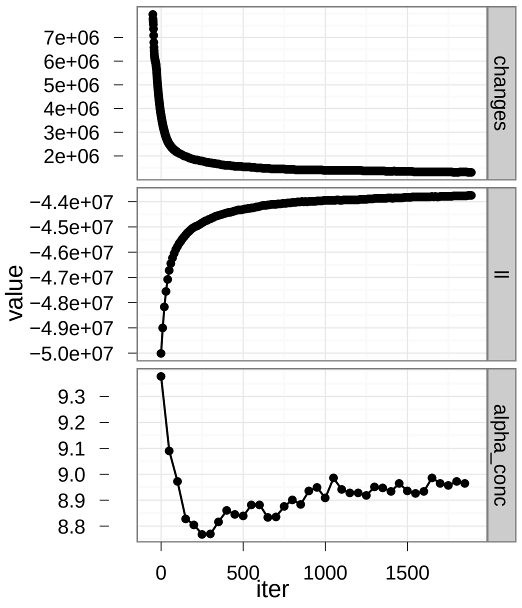
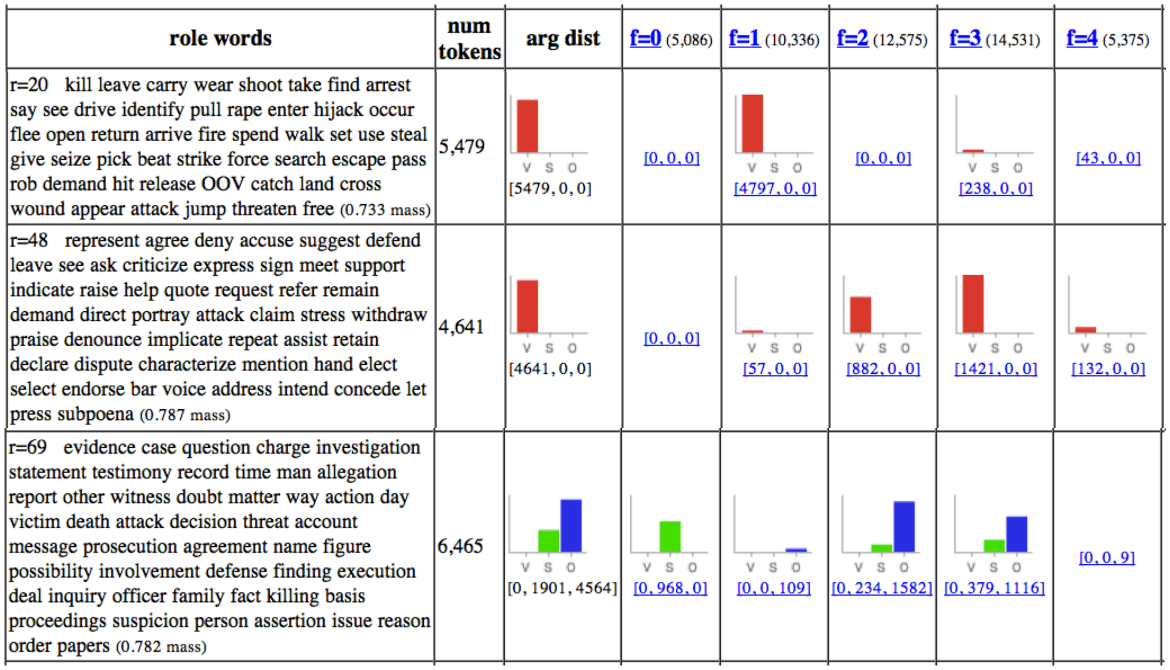
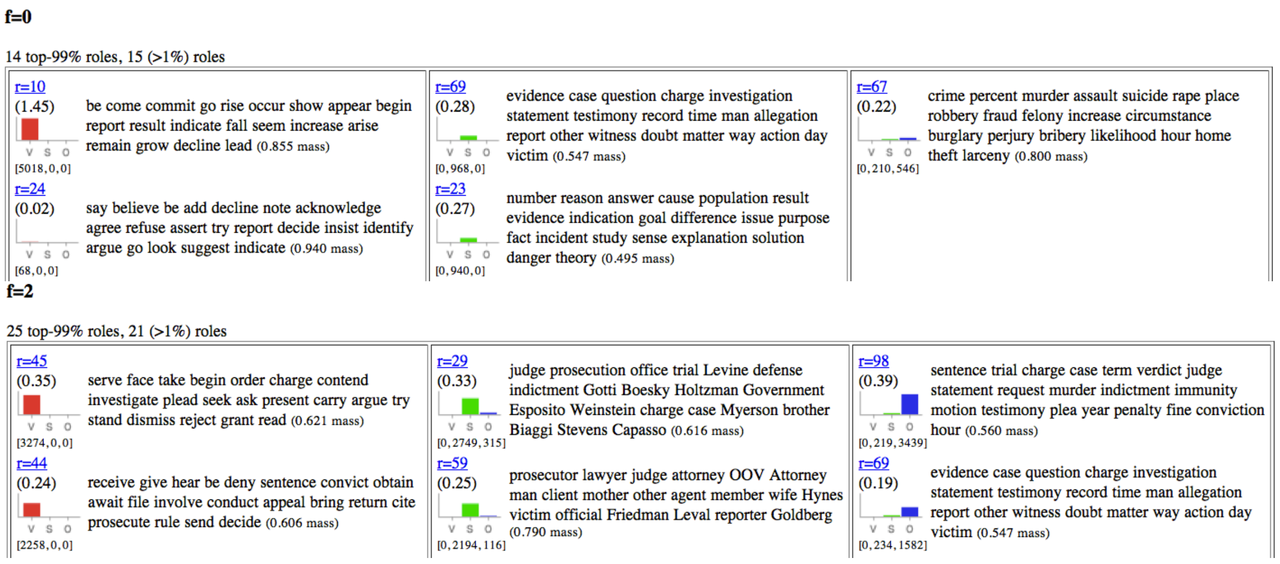


Figure 10: Model 2 trace plots, $K = 10, R = 100$, full dataset.



role words	num tokens	arg dist	f=0 (5,086)	f=1 (10,336)	f=2 (12,575)	f=3 (14,531)	f=4 (5,375)
r=20 kill leave carry wear shoot take find arrest say see drive identify pull rape enter hijack occur flee open return arrive fire spend walk set use steal give seize pick beat strike force search escape pass rob demand hit release OOV catch land cross wound appear attack jump threaten free (0.733 mass)	5,479	V S O [5479, 0, 0]	[0, 0, 0]	V S O [4797, 0, 0]	[0, 0, 0]	V S O [238, 0, 0]	[43, 0, 0]
r=48 represent agree deny accuse suggest defend leave see ask criticize express sign meet support indicate raise help quote request refer remain demand direct portray attack claim stress withdraw praise denounce implicate repeat assist retain declare dispute characterize mention hand elect select endorse bar voice address intend concede let press subpoena (0.787 mass)	4,641	V S O [4641, 0, 0]	[0, 0, 0]	V S O [57, 0, 0]	V S O [882, 0, 0]	V S O [1421, 0, 0]	V S O [132, 0, 0]
r=69 evidence case question charge investigation statement testimony record time man allegation report other witness doubt matter way action day victim death attack decision threat account message prosecution agreement name figure possibility involvement defense finding execution deal inquiry officer family fact killing basis proceedings suspicion person assertion issue reason order papers (0.782 mass)	6,465	V S O [0, 1901, 4564]	V S O [0, 968, 0]	V S O [0, 0, 109]	V S O [0, 234, 1582]	V S O [0, 379, 1116]	[0, 0, 9]

Figure 11: Fragment of a Model 2 role-frame matrix; $K = 10, R = 100$, full dataset. *(0.787 mass)* indicates that thewords shown comprise 0.787 of the total probability mass of that word multinomial. [BTO: TODO massive overhaul, this is a quick-and-dirty screenshot from my HTML inspector—bad. going to all change anyways]



f=0

14 top-99% roles, 15 (>1%) roles

r=10 (1.45) V S O [5018,0,0] be come commit go rise occur show appear begin report result indicate fall seem increase arise remain grow decline lead (0.855 mass)

r=24 (0.02) V S O [68,0,0] say believe be add decline note acknowledge agree refuse assert try report decide insist identify argue go look suggest indicate (0.940 mass)

r=69 (0.28) V S O [0,968,0] evidence case question charge investigation statement testimony record time man allegation report other witness doubt matter way action day victim (0.547 mass)

r=23 (0.27) V S O [0,940,0] number reason answer cause population result evidence indication goal difference issue purpose fact incident study sense explanation solution danger theory (0.495 mass)

r=67 (0.22) V S O [0,210,546] crime percent murder assault suicide rape place robbery fraud felony increase circumstance burglary perjury bribery likelihood hour home theft larceny (0.800 mass)

f=2

25 top-99% roles, 21 (>1%) roles

r=45 (0.35) V S O [3274,0,0] serve face take begin order charge contend investigate plead seek ask present carry argue try stand dismiss reject grant read (0.621 mass)

r=44 (0.24) V S O [2258,0,0] receive give hear be deny sentence convict obtain await file involve conduct appeal bring return cite prosecute rule send decide (0.606 mass)

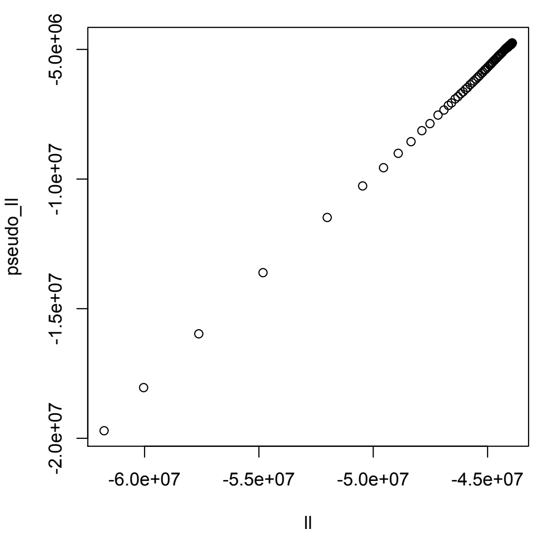
r=29 (0.33) V S O [0,2749,315] judge prosecution office trial Levine defense indictment Gotti Boesky Holtzman Government Esposito Weinstein charge case Myerson brother Biaggi Stevens Capasso (0.616 mass)

r=59 (0.25) V S O [0,2194,116] prosecutor lawyer judge attorney OOV Attorney man client mother other agent member wife Hynes victim official Friedman Leval reporter Goldberg (0.790 mass)

r=98 (0.39) V S O [0,219,3439] sentence trial charge case term verdict judge statement request murder indictment immunity motion testimony plea year penalty fine conviction hour (0.560 mass)

r=69 (0.19) V S O [0,234,1582] evidence case question charge investigation statement testimony record time man allegation report other witness doubt matter way action day victim (0.547 mass)

Figure 12: Two frames from the above model, with their highest-probability roles for each (V,S,O) argument position—these are two of the frames for which role 61 is active. [BTO: TODO this is impossible to read...]



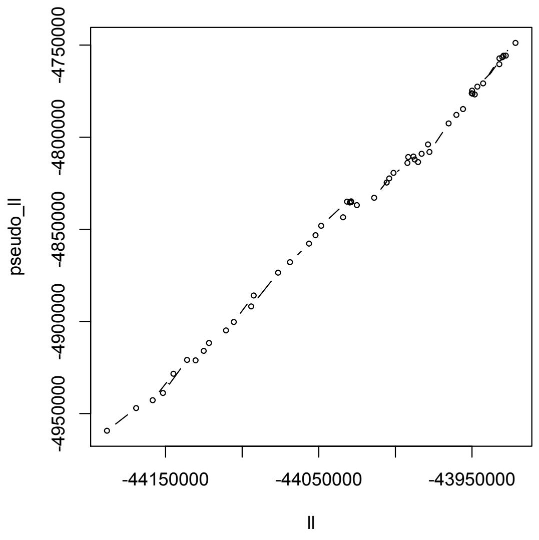


Figure 13: Correlation of running pseudolikelihood (evaluated during Gibbs sampling) to actual likelihood (evaluated exactly via the DCM PMF (section 9)), for one MCMC run (small dataset, Model 1). Left: shown for all iterations where likelihood was evaluated. Right: shown for iterations 500 and later, with lines drawn between successive iterations.