Explorations into Semantic Coherence of Topics in Topic Models for Short Texts (like Twitter)

Shubhomoy Das

1 Preliminaries

1.1 Latent Semantic Analysis (LSA)

Any information retrieval algorithm is faced with two language properties – *synonymy* and *polysemy*. *Synonymy* refers to the property that an object can be referred to using more than one word. For instance, *document* and *file* might both refer to stored files on a computer. *Polysemy* is the property that one word might refer to more than one type of object, e.g. *bank* might refer to a financial institution or a location by the river-side. When a user searches for information using query words, synonymy usually reduces recall while polysemy reduces precision. Both properties reduce the retrieval accuracy.

These problems led to research in identifying semantic relationships between documents and terms. One approach is to cluster documents using distance based methods like k-means. These clustering approaches were limited in capturing the richness of language and while they are computationally efficient, they do not necessarily improve the retrieval accuracy. One of the earliest successful efforts was Latent Semantic Analysis [4] which uses singular value decomposition (SVD) to identify the latent structure. Recall that SVD decomposes a matrix $\mathbf{X}$ in the following way:

$$\mathbf{X} = USD^T \tag{1}$$

where the columns of matrices U and D are orthonormal and the matrix S is diagonal containing the eigen values of $\mathbf{X}$.

Documents are commonly represented as feature vectors where words are features. The value at each feature position is the count of occurrences of the corresponding word in the document. The entire document corpus can be represented as a matrix $\mathbf{X}$ where the rows are the document vectors.

When the document matrix $\mathbf{X}$ is decomposed using SVD, the columns of the orthogonal matrices act as new uncorrelated *factors*. Many of these factors have very small influence and are dropped. The remaining factors can be treated as artificial concepts. The distance between two documents in the space defined by these factors is claimed to be a more reliable estimate of their semantic differences than the distance in the original feature space. The main argument for this is that documents which share frequently co-occurring terms will have similar representation in the transformed space even if they have no terms in common.

Despite its success in clustering documents, LSA lacks a sound statistical foundation. Partly, this is due to the use of L_2-norm or Frobenius norm as the objective function in SVD which implies an additive Gaussian noise assumption on the word counts [10]. This assumption might not be true for multinomial count data such as text. Besides this, LSA also fails to handle polysemy effectively, and moreover, tends to be computationally intensive when the matrix dimensions are high.

1.2 Probabilistic Latent Semantic Analysis (PLSA)

To overcome the limitations of LSA, [10] proposed PLSA with stronger statistical foundations. This assumes a generative model for documents. Assume that there are K unobserved (latent) classes represented by $Z = \{z_1, ..., z_K\}$. These model the general word co-occurrences. Each occurrence of a word $w \in W = \{w_1, ..., w_M\}$ in a document $d \in D = \{d_1, ..., d_N\}$ is associated with one $z \in Z$. The generative model is then defined as:

1. select a document $d \sim P(d)$
2. draw a latent class $z \sim P(z|d)$
3. draw a word $w \sim P(w|z)$

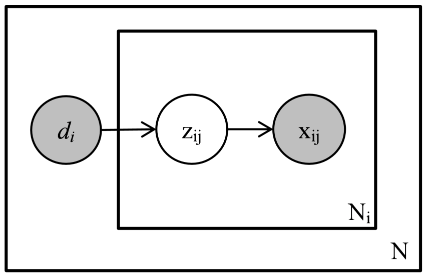


Figure 1: Plate Notation for generative process of PLSA. N is the number of documents, N_i is the number of words in the i-th document. The shaded variables (d_i and x_{ij}) are observed whereas the unshaded ones (z_{ij}) are latent.

Figure 1 illustrates the generative process in *plate notation*. The above process generates an observed pair (d, w), and the latent class z is discarded. The joint probability $P(d, w)$ can also be written as:

$$P(d, w) = \sum_{z \in Z} P(z)P(w|z)P(d|z) \tag{2}$$

We can now express the above equation 2 in matrix notation. Let $U = (P(d_i|z_k))_{i,k}$, $V = (P(w_j|z_k))_{j,k}$, and $S = diag(P(z_k))_k$. The joint probability model over the entire document corpus $\mathbf{P}$ can be written as $\mathbf{P} = USD^T$. This is very similar in form to the SVD decomposition in LSA. However, unlike in LSA which used L_2-norm and Frobenius norm in the objective function, PLSA uses the likelihood function of multinomial sampling and explicitly maximizes the predictive power of the model. This is advantageous because now the factors have clear probabilistic meanings in terms of mixture component distributions.

1.3 Latent Dirichlet Allocation (LDA)

PLSA does not make any distribution assumptions about $P(z|d)$. This makes it difficult to generalize the model to new and unseen documents [19]. Latent Dirichlet Allocation [2] addresses this by introducing a Dirichlet prior on $P(z|d)$. Because of this, LDA might be considered as Bayesian PLSA. The plate notation for the original LDA is shown in Figure 2. The generative process assumption in LDA is:

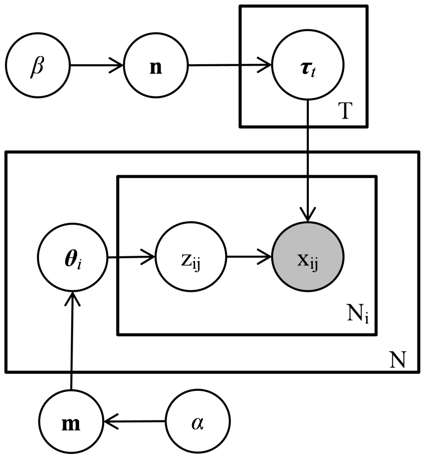


Figure 2: Plate Notation for Generative Process of LDA. N is the number of documents, N_i is the number of words in i-th document, and T is the number of topics.

1. select topics $\mathbf{\Gamma} = \{\tau_1, ..., \tau_T\}$. Each τ_t is a multinomial distribution over V words where V is the size of the vocabulary; $\tau_t \sim Dirichlet(\beta\mathbf{n})$
2. for each document d
   (a) select $\theta_d \sim Dirichlet(\alpha\mathbf{m})$
   (b) for each word w
      i. select $z \sim Multinomial(\theta_d)$
      ii. select $w \sim Multinomial(\tau_z)$

In the Figure 2, **n** is a constant vector of V 1s and **m** is a constant vector of T 1s. α and β are fixed hyper-parameters.

Conceptually, LDA factorizes the document matrix $\mathbf{X}$ as:

$$\mathbf{X}_{[N\times V]} = \mathbf{\Theta}_{[N\times T]}\mathbf{\Gamma}_{[T\times V]} \tag{3}$$

where $\mathbf{\Theta} = \{\theta_1, ..., \theta_N\}$ and $\mathbf{\Gamma} = \{\tau_1, ..., \tau_T\}$.

This is very similar to Equation 1 if we assume that the diagonal matrix S gets absorbed into either U or D. Therefore, LDA might also be considered as a form of matrix factorization that projects documents into a lower dimensional space. The LDA topics are direct counterparts of the orthogonal factors in LSA which represented artificial concepts. However, unlike in LSA, the topics under LDA have nicer probabilistic interpretations and always have positive values. Also, the topics in LDA are not orthogonal.

A popular inference algorithm for computing the parameters Γ and Θ is *Collapsed Gibbs Sampling* [9, 8]. The main posterior that needs to be computed for the Gibbs sampling step is:

$$P(z_i = t|\mathbf{z}_{-i}, \mathbf{w}) \propto \frac{n_{t,-i}^{(w_i)} + \beta}{n_{t,-i}^{(.)} + V\beta} \times \frac{n_{t,-i}^{(d_i)} + \alpha}{n_{.,-i}^{(d_i)} + T\alpha} \tag{4}$$

where the subscript i ranges over all document-word pairs in the corpus. w_i is the feature/word corresponding to the i-th pair. d_i is the document corresponding to the i-th pair. $P(z_i = t|\mathbf{z}_{-i}, \mathbf{w})$ is the probability that the i-th pair is assigned the t-th topic. $\mathbf{z}_{-i}$ refers to all topic assignments excluding the i-th pair. $n_{t,-i}^{(w_i)}$ refers to the number of times the word w_i has been assigned to the

t-th topic excluding the i-th pair. $n^{(\cdot)}_{t,-i}$ is the total number of word assignments to the t-th topic excluding the i-th pair. $n^{(d_i)}_{t,-i}$ is the number of words in document d_i that have been assigned to topic t excluding the i-th pair. $n^{(d_i)}_{\cdot,-i}$ is the total number of words in document d_i excluding the i-th pair.

In collapsed Gibbs sampling we do not need to sample the posteriors of the parameters Θ and Γ. This makes the sampling very efficient. Moreover, the posterior computation requires only simple bookkeeping and is therefore easy to implement. The parameters can be recovered once the MCMC has converged:

$$\tau_{t,w} = \frac{n^{(w)}_t + \beta}{n^{(\cdot)}_t + V\beta} \tag{5}$$

$$\theta_{d,t} = \frac{n^{(d)}_t + \alpha}{n^{(d)}_{\cdot} + T\alpha} \tag{6}$$

In Equation 5, $n^{(w)}_t$ is the number of times word w has been assigned to topic t and $n^{(\cdot)}_t$ is the total number of words assigned to topic t. In Equation 6, $n^{(d)}_t$ is the number of word occurrences assigned to topic t in document d and $n^{(d)}_{\cdot}$ is the total number of words in document d.

2 Modified LDA

We saw in the above section how we could model a corpus by a generative process and then recover the underlying concepts. This is fortunate because a generative process is closer to human intuition than a purely mathematical function like SVD. It is then easier to correct model deficiencies by modifying the generative process.

Although LDA has good predictive power, the topics frequently do not make sense to humans [3]. This problem gets bigger when modeling micro-blogs (like *tweets*) where messages are short and noisy. In larger documents like Wikipedia articles, blog posts etc., the author might be drawing material from various related topics but possibly each sentence or paragraph pertains to a single topic. The document as a whole ends up having a mixture of topics. In the case of short documents like tweets, each message is like a sentence related to a single topic.

Various approaches have been suggested in recent literature to remedy the problem of topic coherence. One approach is to use the original LDA model but aggregate the data in some way, for instance, all messages from one user are combined into one large document [11]. This approach throws out the message-level information which might be valuable. Another approach is to introduce additional structure in the LDA model itself [21]. I take the latter approach in the work presented here.

2.1 Topic per Tweet LDA (TpT-LDA)

The original LDA model treats each document as a mixture of multiple topics. For very short documents like *tweets*, a reasonable assumption might be that each tweet is generated from a single topic. Intuitively, if this were not the case, then its recipients would be easily confused. The mixture-of-topics aspect should now apply at the user level (instead of a tweet) since it is natural to expect a user to tweet on multiple topics. We refer to this model as TpT-LDA in the rest of the paper. TpT-LDA is very similar to [21] with the difference being that [21] adds an additional *junk* topic which we do not. More importantly, we use WordNet in this project to evaluate topic coherence which is different from [21], and in some ways more objective.

The Multinomial Naive Bayes (MNB) model used in text classification [16] is also based on the topic-per-document assumption, but requires supervised training (with labeled data). TpT-LDA might be considered approximately as an unsupervised version of MNB.

The generative process for TpT-LDA is shown below.

1. select topics $\mathbf{\Gamma} = \{\tau_1, ..., \tau_T\}$. Each τ_t is a multinomial distribution over V words where V is the size of the vocabulary; $\tau_t \sim Dirichlet(\beta\mathbf{n})$
2. for each user u_i
   (a) select $\theta_i \sim Dirichlet(\alpha\mathbf{m})$
   (b) for the j-th tweet of u_i
      i. select a topic $z_{ij} \sim Multinomial(\theta_i)$
      ii. for each word x_{ijm} in the tweet, select $x_{ijm} \sim Multinomial(\tau_{z_{ij}})$

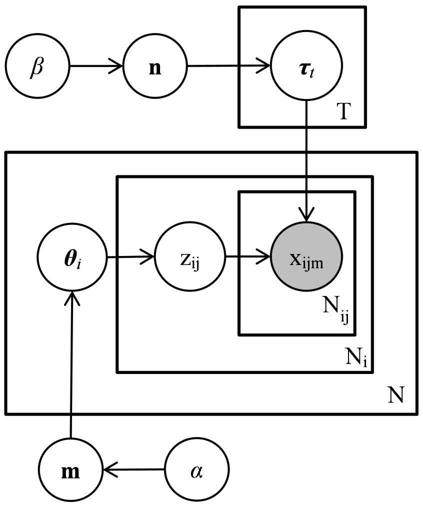


Figure 3: Generative Process for TpT-LDA. N is the number of users, N_i is the number of tweets by i-th user, N_{ij} is the number of words in tweet, and T is the number of topics.

Figure 3 shows the generative process in plate notation. Since the inference technique is not the focus of this paper, we defer the derivations to a supplementary document and present here only the main posterior distribution (Equation 7) that needs to be sampled during MCMC.

$$P(z_i = t|\mathbf{z}_{-i}, \mathbf{X}) \propto \frac{\prod_{m \in \mathbf{X}_i} \prod_{j=0}^{f_m} (n_{t,-i}^{(m)} + \beta + j)}{\prod_{j=0}^{length(\mathbf{X}_i)} (\sum_{m=1}^{V} (n_{t,-i}^{(m)} + \beta + j))} \times (n_{n,-i}^{(t)} + \alpha) \tag{7}$$

The parameters $\mathbf{\Gamma}$ and $\mathbf{\Theta}$ can be computed in a similar way as in Equations 5 and 6.

$$\tau_{t,m} = \frac{n_t^{(m)} + \beta}{n_t^{(.)} + V\beta} \tag{8}$$

$$\theta_{n,t} = \frac{n_n^{(t)} + \alpha}{n_n^{(.)} + T\alpha} \tag{9}$$

Pages 3 to 5 of the supplementary materials document explain the terms appearing in the above Equations 7 through 9. The Gibbs sampling is performed as shown in Algorithm 1.

```
Initialize all $z_i$s to random values;
for each complete pass through all documents in corpus do
    for each document $d$ do
        Exclude $d$ and adjust the book-keeping structures;
        Use Equation 7 to compute the likelihood of $d$ belonging to a topic $t$;
        Sample the new topic for $d$ based on the likelihoods computed;
        Assign $d$ to the new topic and adjust the book-keeping structures;
    end
end
```

Algorithm 1: Gibbs Sampling for TpT-LDA

2.2 Grouped Users and Topic per Tweet LDA (GTpT-LDA)

In our second model, we impose another hierarchy – the user group. It is reasonable to expect that users who share similar personal interests would also have similar topic proportions in their tweets. If we can group users on the basis of topic proportions, that information could then be used to (say) advertise similar products to all users within a group. It is also possible that users who frequently communicate with each other share similar topic proportions. If we already knew the user communities, we could then fix the group memberships and infer the topic proportions within groups. Users often refer to each other using *hashtags* or *handles* in their tweets. The resulting social network graph generated from these references might be used to discover online user communities [13]. The plate notation is shown in Figure 4. The generative process for GTpT-LDA is:

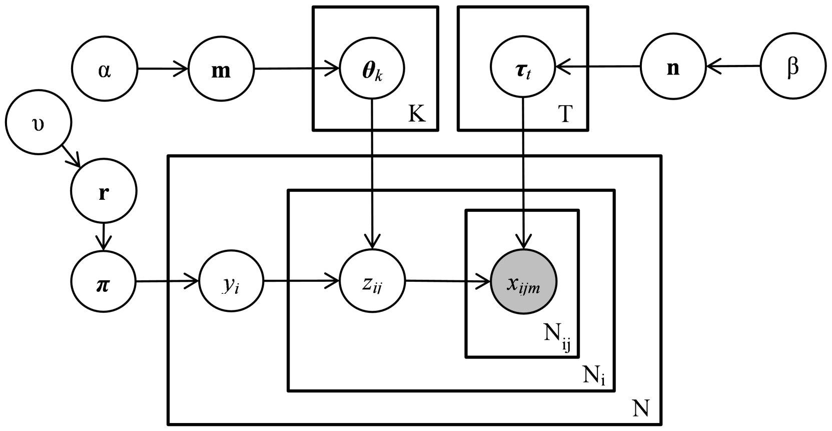


Figure 4: Generative Process for GTpT-LDA. N is the number of users, N_i is the number of tweets by i-th user, N_{ij} is the number of words in tweet, T is the number of topics, and K is the number of user groups.

1. select group probability distribution $\pi = \{\pi_1, ..., \pi_K\}$ where K is the pre-determined number of user groups and $\pi \sim Dirichlet(\nu\mathbf{r})$; $\mathbf{r}$ is a $K \times 1$ vector of 1s; ν is a predetermined constant.

2. select topics $\mathbf{\Gamma} = \{\tau_1, ..., \tau_T\}$. Each τ_t is a multinomial distribution over V words where V is the size of the vocabulary; $\tau_t \sim Dirichlet(\beta\mathbf{n})$
3. select $\mathbf{\Theta} = \{\theta_k \sim Dirichlet(\alpha\mathbf{m})\}_{k=1}^{K}$; each θ_k is a $K \times 1$ vector of topic distribution for the k-th user group.
4. for each user u_i
   (a) select group membership $y_i \sim Multinomial(\pi)$
   (b) for the j-th tweet of u_i
      i. select a topic $z_{ij} \sim Multinomial(\theta_{y_i})$
      ii. for each word x_{ijm} in the tweet, select $x_{ijm} \sim Multinomial(\tau_{z_{ij}})$

We need the following posterior probabilities for Gibbs sampling in GTpT-LDA:

$$P(y_i = k|\mathbf{Y}_{-i}, \mathbf{z}_i, \mathbf{Z}_{-i}, \mathbf{X}) \propto \frac{\prod_{t=1}^{T} \prod_{j=0}^{m_{y_i,i}^{(t)}-1} (m_{k,-i}^{(t)} + \alpha + j)}{\prod_{j=0}^{(\sum_{t'=1}^{T} m_{y_i,i}^{(t')})-1} (\sum_{t=1}^{T} (m_{k,-i}^{(t)} + \alpha + j))} \times (n_{-i}^{(k)} + \nu) \quad (10)$$

$$P(z_i = t|y_i, \mathbf{Z}_{-i}^{y_i}, \mathbf{X}^{y_i}) \propto \frac{\prod_{m\in\mathbf{X}_i} \prod_{j=0}^{f_m} (n_{t,-i}^{(y_i,m)} + \beta + j)}{\prod_{j=0}^{length(\mathbf{X}_i)} (\sum_{m=1}^{V} (n_{t,-i}^{(y_i,m)} + \beta + j))} \times (m_{y_i,-i}^{(t)} + \alpha) \quad (11)$$

The parameters π, $\mathbf{\Gamma}$ and $\mathbf{\Theta}$ can be computed as:

$$\tau_{t,m} = \frac{n_t^{(m)} + \beta}{n_t^{(\cdot)} + V\beta} \quad (12)$$

$$\theta_k^t = \frac{m_k^t + \alpha}{m_k^{(\cdot)} + T\alpha} \quad (13)$$

$$\pi_k = \frac{n^{(k)} + \nu}{n^{(\cdot)} + K\nu} \quad (14)$$

Pages 9 and 10 of the supplementary materials document explain the terms appearing in the above Equations 10 through 14. Please note that y_i in Equation 10 is for *user i* whereas z_i in Equation 11 is for *document i*. This might seem confusing, but the supplementary text clarifies this distinction. In this project we fix the user group memberships and only sample the document topics. Hence, the Gibbs sampling procedure is very similar to that of TpT-LDA (Algorithm 1).

3 WordNet

The main purpose of our modifications to LDA has been to identify themes in a corpus which are easily interpretable by humans. It seems reasonable to expect that to be interpretable, the themes – represented by topics – should show a high degree of *semantic coherence* and that the most frequently co-occurring words in coherent topics should also be closely *related* to each other in terms of their definitions in a thesaurus. A precise definition for *coherence* for topics does not exist yet (though current research [15] suggests potential alternatives) and it therefore is difficult to measure. One alternative is to measure *relatedness* and use that as a proxy for *coherence*. *Relatedness* between any two words might be obtained either from some external data source or from human experts [3].

The latter option is costly and prone to subjective opinions; therefore we use WordNet in this project to compute the relatedness scores.

WordNet [5] is an online lexical database of English and resembles a thesaurus. It organizes words into sets of cognitive synonyms (synsets) in a way that facilitates computation of semantic relations between words. It is popularly used in the linguistic and natural language processing community.

Every word could have more than one form – noun, verb, adverb, or adjective. Within each form, a word usually has more than one definition based on usage. These different definitions are referred to as *senses*. To compute a relatedness score between two words, we exhaustively query the relatedness scores across all combinations of their forms and senses and then take the average of these scores to get a final relatedness score. As a reasonable strategy to avoid too many online requests to WordNet, we consider only the top six senses of each word form.

There are multiple algorithms for computing relatedness between words and several of these have been implemented on top of WordNet [15]. These algorithms are available through programmatic APIs which makes WordNet convenient for our purpose. The five algorithms I have used in this project are briefly explained below.

Path

WordNet structures *synsets* in hypernym/hyponym (nouns) or hypernym/troponym (verbs) hierarchies. The *path* distance between two words is the number of nodes visited along the shortest path from one word to the other via the hypernym hierarchy.

Lin

Lin [14] similarity is based on an information-theoretic approach that scales the Information Content of each node by the information content of their least common subsumer (LCS). LCS is defined as the deepest node in the WordNet hierarchy that subsumes synsets of both the words which are being compared.

Res

Res [17] computes the path length using weighted edges between nodes. The edges are weighted by their frequency of use in textual corpora.

Lesk

Lesk [12, 1] computes the similarity between two words by using the lexical overlap in their WordNet sense *glosses*.

Vector

Vector [18] creates a context vector to describe the context in which a particular word sense appears. The *context* is based on the thesaurus definition and related words. The similarity between two words is then computed as the cosine similarity between their context vectors.

4 Experiments

4.1 Synthetic Data

Users	Topic 1 Proportion	Topic 2 Proportion	Topic 3 Proportion
User 1 ... User 10	0.3	0.55	0.15
User 10 ... User 20	0.55	0.30	0.15
User 20 ... User 30	0.15	0.20	0.65

Table 1: Topic Proportions in Synthetic Data

A synthetic dataset was used to test the correctness of inference using collapsed Gibbs Sampling. Three distinct topics were defined based on 50 features (words) as shown in Figure 5. Each topic defines a multinomial probability distribution over all 50 features. Under *Topic 1*, feature 1 has the highest probability of occurrence and the probability decreases exponentially till feature 50. Similarly, *Topic 2* and *Topic 3* were setup to have distinct distributions. Next, 30 hypothetical users were defined with topic distributions shown in Table 1. It can be seen that these users are clustered into three distinct groups based on their topic distributions. For each user, we generated 300 short documents (with length distributed as Poisson($\lambda = 5$)) by simulating the generative process defined in Section 3.1. This resulted in 9000 total *documents*. All three LDA models were configured to use the correct number of topics – three. GTpT-LDA was also configured to use three user groups. With these 'ground-truth' configurations, all models identified the feature distributions in topics with high accuracy. GTpT-LDA also grouped the users correctly. The MCMC converged after 15 iterations through the data. Of course, in most real application it will be hard to know the real number of topics or user groups and the models could be sensitive to these settings.

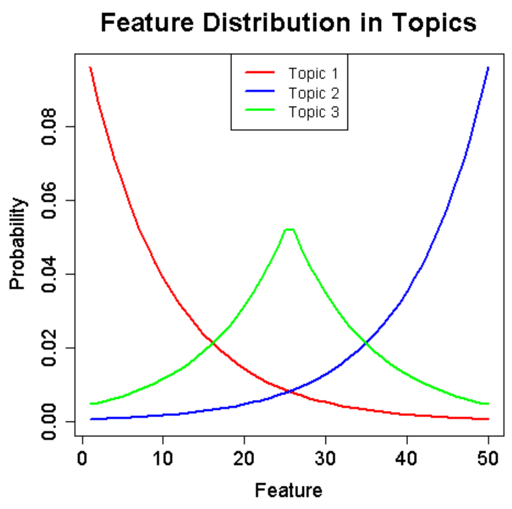


Figure 5: Topics in Synthetic Dataset

4.2 Twitter Data

The Twitter data used in this project has been made freely available from [7] primarily for research in sentiment analysis [6]. Therefore, the dataset has a bias towards some subjects/topics that invoke strong opinions. The biased nature of data collection might also hide the social network connections making it impossible to identify 'ground-truth' user-groups. In my opinion, despite these issues we should still be able to reasonably compare the simple LDA and TpT-LDA. I have made an effort to identify the user groups by mining the user communication links. My objective is to assign users appropriately to groups and let that guide inference in GTpT-LDA. However, as we will see later, the results for GTpT-LDA are not encouraging.

As part of pre-processing, I removed stop-words, words that occur less than 50 times in the corpus, and some of the most frequent words which appear to be junk. Tweets contain *hashtags* and *handles*. These should not be made part of the vocabulary and therefore were removed from the message body. The *handles* were used to find connections between users. Using these connections 333 user groups were identified. The group sizes vary from around 150 to 360 members. All users not belonging to any group were removed. The resulting corpus has 78590 users, 437088 tweets, and a vocabulary of 4904 unique words. Also, instead of using a hold-out set for doing model selection to pick the correct number of topics, I have followed the common practice of using 100 topics.

For GTpT-LDA we assume that user groups are known (as mentioned previously) and therefore do not sample the groups in MCMC iterations – i.e., we ignore Equation 10. For a corpus of this size, it is common (from published literature) to run the MCMC for 1000 iterations. Ideally, we should stop the MCMC only when it has converged. One way to check convergence is by computing the likelihood on a hold-out set. However, this computation tends to be complicated and inaccurate [20]. Therefore I have skipped this computation and instead, trained all models for 4000 iterations through the complete data.

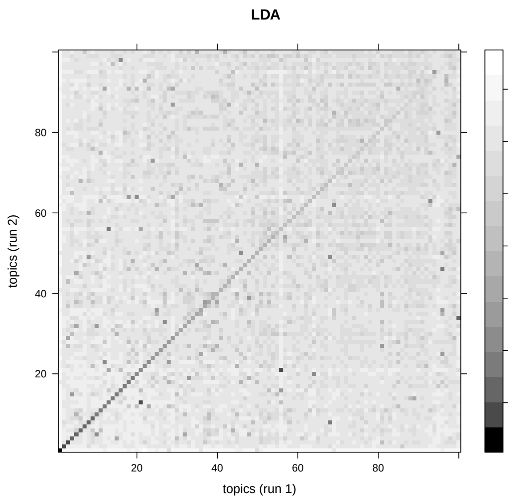
4.3 Results and Discussion

Our first concern is to check the statibility of the inferred parameters. If the inference is stable, then we should see very similar topics every time we re-run the inference with a different set of random initializations. Therefore, we ran MCMC twice for all our algorithms and then computed the similarity between topics using symmetrized KL-divergence. Assume that $\mathbf{P} = \{\mathbf{p_1}, ..., \mathbf{p_T}\}$ and $\mathbf{Q} = \{\mathbf{q_1}, ..., \mathbf{q_T}\}$ are topics inferred in the first and second run respectively for an algorithm. Then, the symmetrized KL divergence is computed as shown in Equation 15.

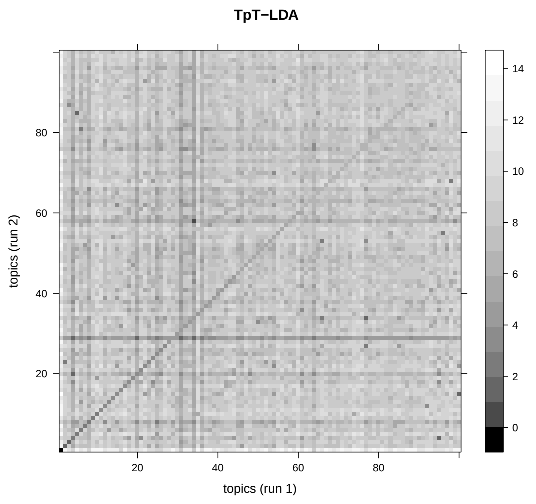
$$KL(\mathbf{p_i}, \mathbf{q_j}) = \frac{1}{2}\sum_{v=1}^{V}(p_{iv}log(p_{iv}) - p_{iv}log(q_{jv})) + \frac{1}{2}\sum_{v=1}^{V}(q_{jv}log(q_{jv}) - q_{jv}log(p_{iv})) \tag{15}$$

To visually compare the topics, we pair each topic from the first run with an unpaired topic from the second run to which it is most similar. We then order these pairs according to decreasing similarities and get a natural ordering over topics using which we plot all pairwise similarities as a heatmap shown in Figure 6.

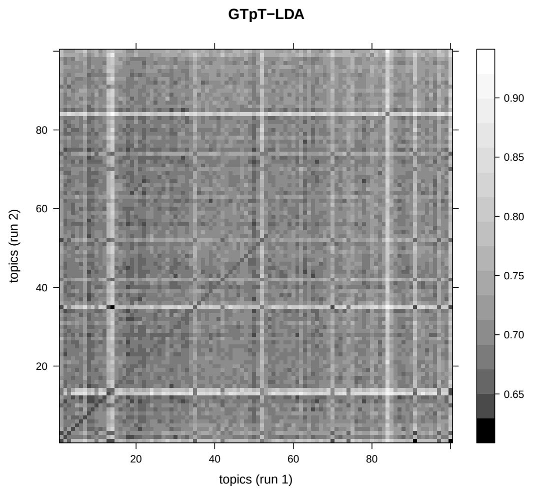
In Figure 6 the darker shades correspond to higher similarity whereas the lighter shades correspond to higher dissimilarity. The original LDA topics appear to be more stable. My hypothesis is that TpT-LDA and GTpT-LDA are much more prone to local optima because the additional grouping structures give rise to non-linearities. This problem is especially acute in GTpT-LDA as can be seen in Figure 6 where the scale of KL-divergence varies from 0 to 1 only, thus indicating that the topics are mostly random. To avoid this problem, we should probably have better initialization of the topics – maybe first initialize the topics from simple LDA. It should be noted that the groups we have assigned to users might not be ideal and that further confuses inference.



(a) Original LDA



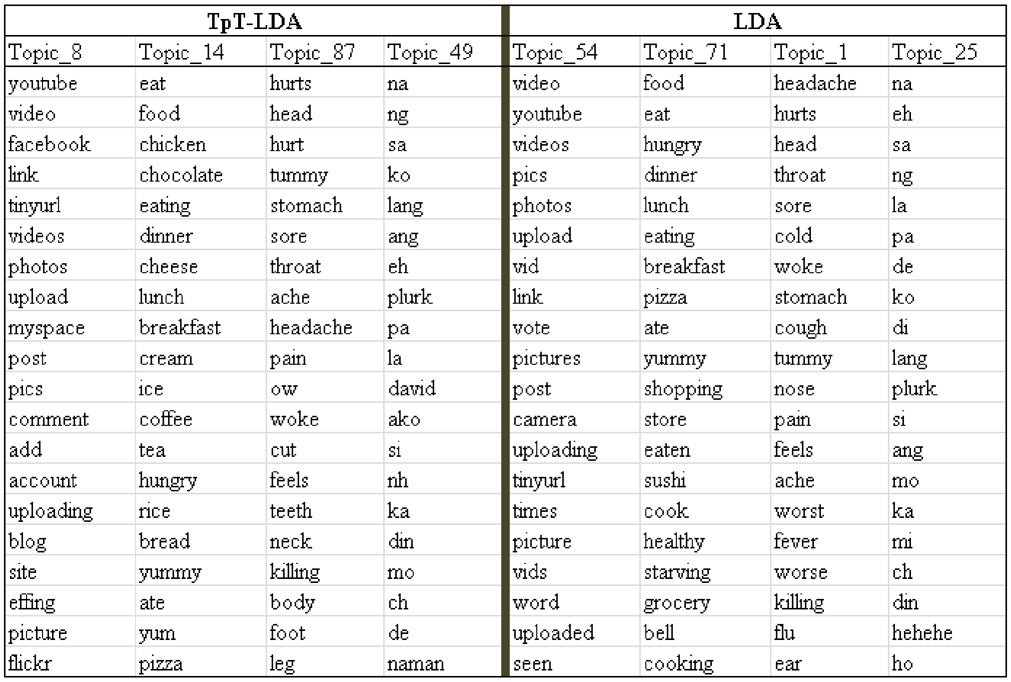
(b) TpT-LDA



(c) GTpT-LDA

Figure 6: Topic stability across two randomly initialized MCMC runs for parameter inference. Darker shades represent more similarity between topics.

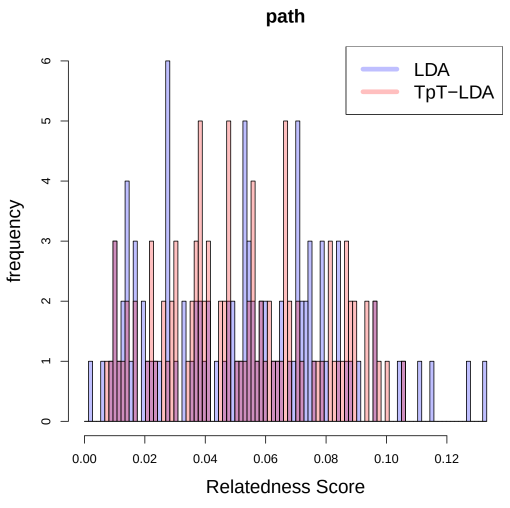
Since the GTpT-LDA results are highly unreliable, I have only compared the semantic coherence of topics in the original LDA with that of TpT-LDA. Figure 7 shows some similar topics detected by LDA and TpT-LDA. Zhao et al. [21] reported that using a model similar to TpT-LDA, they could improve the topic coherence. In our experiments however, we do not observe such an improvement. Following the convention that the top 10 words within a topic are considered representative of its coherence, I compute the average relatedness score for each topic using all pair-wise combinations of the top 10 words in that topic. Since I have configured each model to have 100 topics, I get 100 relatedness scores for each model. Due to lack of space, I show the distributions of scores under *path* and *lesk* only in Figure 8. Using the Wilcoxon rank-sum test for unpaired data, I did not find any significant difference in relatedness scores (as proxy for topic coherence) between LDA and TpT-LDA under any of the five WordNet measures mentioned above.



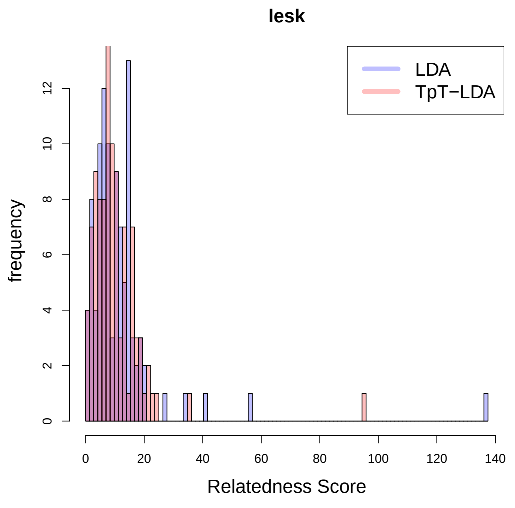
TpT-LDA				LDA			
Topic_8	Topic_14	Topic_87	Topic_49	Topic_54	Topic_71	Topic_1	Topic_25
youtube	eat	hurts	na	video	food	headache	na
video	food	head	ng	youtube	eat	hurts	eh
facebook	chicken	hurt	sa	videos	hungry	head	sa
link	chocolate	tummy	ko	pics	dinner	throat	ng
tinyurl	eating	stomach	lang	photos	lunch	sore	la
videos	dinner	sore	ang	upload	eating	cold	pa
photos	cheese	throat	eh	vid	breakfast	woke	de
upload	lunch	ache	plurk	link	pizza	stomach	ko
myspace	breakfast	headache	pa	vote	ate	cough	di
post	cream	pain	la	pictures	yummy	tummy	lang
pics	ice	ow	david	post	shopping	nose	plurk
comment	coffee	woke	ako	camera	store	pain	si
add	tea	cut	si	uploading	eaten	feels	ang
account	hungry	feels	nh	tinyurl	sushi	ache	mo
uploading	rice	teeth	ka	times	cook	worst	ka
blog	bread	neck	din	picture	healthy	fever	mi
site	yummy	killing	mo	vids	starving	worse	ch
effing	ate	body	ch	word	grocery	killing	din
picture	yum	foot	de	uploaded	bell	flu	hehehe
flickr	pizza	leg	naman	seen	cooking	ear	ho

(a) Path

Figure 7: Top words for some topics detected by LDA and TpT-LDA. Some topics are junk like Topic_49 in TpT-LDA and Topic_25 in LDA.



(a) Path



(b) Lesk

Figure 8: Relatedness score distribution – higher score is better

5 Conclusion

To make sense of the huge database of twitter messages it would help to improve topic coherence in LDA models as they are scalable and frequently useful in text mining applications. We attempted to do this by extending the LDA. Although previously published literature has shown improvements to topic coherence using a model very similar to TpT-LDA, we did not see such improvements. It might be that improvement can be detected under human evaluation and that the WordNet database has limitations. Also, we suspect that introducing grouping into the model risks running into local optima more often and therefore we need to investigate better ways of initialization in future work.

References

[1] S. Banarjee and T. Pedersen. An adapted lesk algorithm for word sense disambiguation using WordNet. In *CICLing-2002*, 2002.

[2] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[3] Jonathan Chang, Jordan L Boyd-Graber, Sean Gerrish, Chong Wang, and David M Blei. Reading tea leaves: How humans interpret topic models. In *Neural Information Processing Systems*, volume 22, pages 288–296, 2009.

[4] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[5] Christiane Fellbaum. Wordnet: An electronic lexical database. `http://wordnet.princeton.edu`, 1998.

[6] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *Processing*, pages 1–6, 2009.

[7] Alec Go, Richa Bhayani, and Lei Huang. Sentiment140 website. `http://help.sentiment140.com/for-students`, 2009.

[8] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235, April 2004.

[9] Gregor Heinrich. Parameter estimation for text analysis. Technical report, 2004.

[10] Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM.

[11] Liangjie Hong and Brian D. Davison. Empirical study of topic modeling in twitter. In *Proceedings of the First Workshop on Social Media Analytics*, SOMA '10, pages 80–88, New York, NY, USA, 2010. ACM.

[12] M. Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of SIGDOC*, pages 24–26, 1986.

[13] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 695–704, New York, NY, USA, 2008. ACM.

[14] D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of COLING/ACL-98*, pages 768–774, Montreal, Canada, 1998.

[15] David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. Automatic evaluation of topic coherence. In *Human Language Technologies*, HLT '10, pages 100–108, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[16] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Mach. Learn.*, 39(2-3):103–134, May 2000.

[17] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, pages 448–453, 1995.

[18] Hinrich Schtze. Automatic word sense discrimination. *Journal of Computational Linguistics*, 24:97–123, 1998.

[19] Mark Steyvers and Tom Griffiths. *Probabilistic Topic Models*. Lawrence Erlbaum Associates, 2007.

[20] Hanna M. Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. Evaluation methods for topic models. In *ICML*, ICML '09, pages 1105–1112, New York, NY, USA, 2009. ACM.

[21] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. Comparing twitter and traditional media using topic models. In *Proceedings of the 33rd European Conference on Advances in Information Retrieval*, ECIR'11, pages 338–349, Berlin, Heidelberg, 2011. Springer-Verlag.