CS388 NLP Project Report

Text Mining for Hidden Relations and Trending

C. Vic Hu
vic@cvhu.org

Ali Unwala
aliunwala@gmail.com

***Abstract*—The main contribution of this paper has two folds. First, we formalized an illustration method to visualize topic trending on a bidirectional spanning tree, which gives a meaningful intuition to discover how hidden thematic structures in large archive of text documents change, merge and split over time. Second, we proposed an algorithm, Thematic Particle Clustering, that combines probabilistic sampling, clustering and gradient descent methods to predict upcoming topics based on a sequence of history topics. The effectiveness of our methods is demonstrated through a collection of 10,000 patent data in the field of robotics spanning over 30 years.**

I. Introduction

In this paper we descibe a data visualizaion technique and and topic modeling method. We started by choosing to work with US Patent dataset. We parsed 10,000 patents using the words ”robot” and ”robotic” over 31 years. These patents were then passed to Mallet [18] one year at a time. Mallet took in the patent data and returned the topic information for each year.

In this paper we have two contributions. The first contribution is visualizing topics as they changed over time. Topics intrinsically create an twisted structure over the course of time. To solve this we made an visualization technique for topics we call a Tree Convergence Graph (TCG). A TCG shows topics converging from year to year and new topics forming over large periods of time. Our visualization technique superimposes topics from year to year until they converge creating an easily parseable tree structure. We found this style of visualization novel and immensely useful in understanding topic relationships from year to year as they progressed.

The second contribution of this paper was a new topic modeling method called Thematic Particle Clustering (TPC). TPC treats each set of words in a topic as a vector. This vector is replicated into particles which are biased by the words counts for that year. We then choose the most relevant particles for a year and then perform error analysis. The intuition for this idea is that we are interested in the topics that have the highest word counts for a year and these topics will have lower error from year to year.

GIVE A OVERVIEW OF ALL SECTIONS

II. Background

Given a collection of text-based patent documents, one intuitive idea to find out trending patterns is to examine the underlying thematic structures hidden in the text. Based on the vocabulary distribution, we want to know what the intrinsic topics are implied in the given context. One common way to do exactly that is the Probabilistic Topic Models formalized by Blei et al. [1]

A. Probabilistic Topic Models

The main objective of topic modeling is to automatically discover the unobserved hidden structures–the topics, per-document topic distributions, and the per-document per-word topic assignments, while a collection of text documents is the only observable variables.

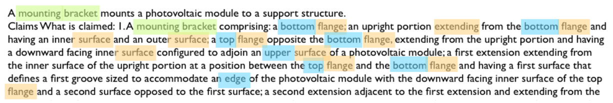

A mounting bracket mounts a photovoltaic module to a support structure.
Claims What is claimed: 1. A mounting bracket comprising: a bottom flange; an upright portion extending from the bottom flange and having an inner surface and an outer surface; a top flange opposite the bottom flange, extending from the upright portion and having a downward facing inner surface configured to adjoin an upper surface of a photovoltaic module; a first extension extending from the inner surface of the upright portion at a position between the top flange and the bottom flange and having a first surface that defines a first groove sized to accommodate an edge of the photovoltaic module with the downward facing inner surface of the top flange and a second surface opposed to the first surface; a second extension adjacent to the first extension and extending from the

Fig. 1: A sample patent document (partial)

For example in Fig. 1, we have annotated a selection of words, with topics distinguished by colors. For the orange topic, we get words like flange, surface and extending, which could be interpreted as the attachment of hardware components. Similarly, the blue and green topics could be translated into topics about installation and mounting respectively. By looking at the text, most human being with common comprehensibility could easily tell what a patent data like Fig. 1 is about, and accordingly highlight the relevant keywords that compose such topics.

Nonetheless, the efficiency and accuracy of human labors don’t scale up easily when the size or complexity of these patent documents increases. The objective of probabilistic topic modeling is to automate this inference process and to provide hidden insights and meaningful intelligence of big data. If we are able to successfully construct a probable thematic structure from a large archive of text data for each time slice in a sequence, we could presumably infer how these topics inherit or inspire each other, and most excitingly, predict the most likely topics in the future.

B. Latent Dirichlet Allocation

Latent Dirichlet allocation, or LDA, is the simplest topic model [2] that assigns each word in the documents a distribution over a fixed number of topics. Instead of having a hard boundary between topic collections, LDA provides a distribution of topics per document, giving the likelihood of a mixed proportion of topic assignments. Namely, all text documents share the same set of topic collection but with different proportions to each topic. For instance in Fig. 2, although there are $K = 100$ topics overall, only a few topics were actually activated.

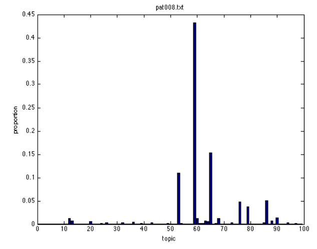


Fig. 2: A sample topic proportion of a patent

$\beta_k,\ k = 1 \cdots K$	The K topics, represented by a distribution over words.
$\theta_d,\ d = 1 \cdots D$	Topic proportions for document d, where $\theta_{d,k}$ is the topic proportion of topic k for document d.
$z_d,\ d = 1 \cdots D$	Topic assignments for document d, where $z_{d,n}$ is the topic assignment for the n-th word in document d.
$w_d,\ d = 1 \cdots D$	The observed words for document d, where $w_{d,n}$ is the n-th word in document d.

TABLE I: Topic modeling notations

To build the generative probabilistic model, we compute the joint distribution and use it to estimate the posterior probability. With the notation specified in Table. I, the LDA generative process can be formalized as the following joint probability of both hidden and observed random variables:

$$
\begin{aligned}
&p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) \\
&= \prod_{i=1}^{K} p(\beta_i) \prod_{d=1}^{D} p(\theta_d) \left(\prod_{n=1}^{N} p(z_{d,n}|\theta_d) p(w_{d,n}|\beta_{1:K}, z_{d,n}) \right)
\end{aligned}
$$

which can also be expressed as a graphical model:

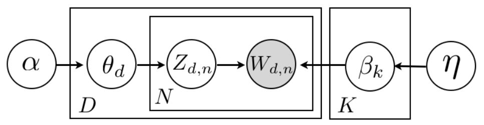


Fig. 3: LDA graphical model. Nodes represent variables, while edges indicate the dependency relations. The shaded node is the only observed variable (document words), and all others are the hidden variables. The D plate denotes the replicated variables product over D documents, while the N plate denotes replication over N words in each document.

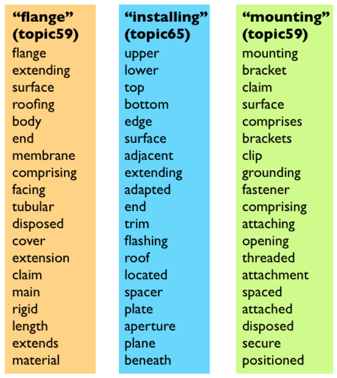
Note that there are several conditional dependencies implied in the graphical models, which reflects the main principles of how LDA "think" the documents are generated:

1) Randomly pick a distribution θ_d over topics.
2) For each word in the document
    a) Randomly choose a topic from the previously-chosen distribution $\theta_{d,n}$.
    b) Randomly choose a word from the corresponding distribution $Z_{d,n}$.

Assuming this generative process is how our documents are created, now LDA uses the graphical model in Fig. 3 to infer the posterior probability of the hidden structures given our observable:

$$
p(\beta_{1:K}, \theta_{1:D}, z_{1:D}|w_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})}
$$

The computation of possible topic structures is often intractable and the posterior distribution can only be approximated in most cases. To form an approximation algorithm, topic modeling can generally be categorized as sampling-based algorithms and variational algorithms. The most popular sampling method for topic modeling is Gibbs sampling, which introduces a sequence of random variables to construct a Markov chain and collects samples from the limiting distribution to estimate the posterior. Instead of using samples to approximate the posterior, variational methods find the closest parameterized distribution candidate by solving optimization problems [2] [3].



"flange" (topic59)	"installing" (topic65)	"mounting" (topic59)
flange	upper	mounting
extending	lower	bracket
surface	top	claim
roofing	bottom	surface
body	edge	comprises
end	surface	brackets
membrane	adjacent	clip
comprising	extending	grounding
facing	adapted	fastener
tubular	end	comprising
disposed	trim	attaching
cover	flashing	opening
extension	roof	threaded
claim	located	attachment
main	spacer	spaced
rigid	plate	attached
length	aperture	disposed
extends	plane	secure
material	beneath	positioned

Fig. 4: The top 3 topics of a sample patent

C. *Limitations & Potential Improvements*

Although LDA provides a powerful perspective to browsing and interpreting the implicit topic structures in our patent corpus, there are a few limitations it imposes against further discoveries. An extensive amount of research has been focused on relaxing some of the assumptions made by LDA to make it more flexible and suitable for various adaptations in more sophisticated context.

LDA is essentially a bag-of-words probabilistic model. Namely, it constructs a word-frequency vector for each document but disregards the word ordering and the neighboring context. Although this assumption looses the syntactic information and sometimes seems unrealistic when processing natural language, it is usually good enough when capturing the document semantics and simplifying hidden structural inferences. Nonetheless, for more sophisticated tasks such as language generation or writing style modeling, the bag-of-words assumption is apparently insufficient and needs to be relaxed. In these cases, there are variants of topic models that generate topic words conditioned on the previous word [4], or switches between LDA and hidden Markov models (HMM) [5].

The LDA graphical model in Fig. 3 is invariant to the ordering of our patent documents, which could be inappropriate if the hidden thematic structure is actually dependent on sequential information such as years published, which is typical in document collections spanning years, decades or centuries. To discover how the topics change over time, the dynamic topic model [6] treats topics as a sequence of distributions over words and tracks how they change over time.

In either LDA or more sophisticated dynamic topic models [6], the number of topics $\beta_{1:K}$ is determined manually and assumed to be fixed. One elegant approach provided by the Bayesian nonparametric topic model [7] is to find a hierarchical tree of topics, in which new documents can now imply previously undiscovered topics.

To include additional attribute information associated with the documents such as authorships, titles, geolocation, citations and many others, an active branch of research has been performed to incorporate meta-data in topic models. The author-topic model [8] associates author similarity based on their topic proportions, the relational topic model [9] assumes document links are dependent on their topic proportion distances, and more general purpose methods such as Dirichlet-multinomial regression models [10] and supervised topic models [11].

Many other extensions of LDA are available, including the correlated topic model [12], pachinko allocation machine, [13], spherical topic model [14], sparse topic models [15] and bursty topic models [16].

III. Problem Definition and Algorithm

A. *Task Definition*

B. *Tree Convergence Graph*

Before we were able to parse the topic models we needed to be able to understand the data we were looking at. Our TCG algorithm followed the following steps

```
array paths;
foreach year:
    foreach topic node:
        paths = print out path
        thorough remaining years
        of current topic
    end foreach topic node
end foreach year

foreach path in paths:
    path = merge topics with
    same converged nodes
end foreach path in paths

plot(paths)
```

C. *Thematic Particle Clustering*

Built on top the results obtained from the LDA topic models, our Thematic Particle Clustering (TPC) algorithm aims to make topic predictions for the upcoming year based on what it has seen in the past. Our goal is to formalize a set of particles $\mathbf{w}_{1:N}$ inferred from the topics from previous years, cluster them and use the results to describe what we think the upcoming topics $\beta_{1:K}$ will be. Before jumping into the details of TPC, let's go over some of the fundamental concepts and definitions we will use.

1) Distance Functions: The idea of a particle is essentially a sampled instance of the topic distribution over a time sequence, represented by a vector $\mathbf{w}_i \in \mathbb{R}^{|V|},\ i \in \{1, \cdots, N\}$, where $|V|$ is the total vocabulary size of all the topic words appeared. Since one of our intermediate objectives is to formalize clusters between these particles, we need to first define how we will measure the similarity or distance between any pair of particles $\mathbf{w}_i,\ \mathbf{w}_j$.

Minkowski:

$$d = \sqrt[p]{\sum_{k=1}^{|V|} |w_{ik} - w_{jk}|^p}$$

Note that when $p = 1$, the Minkowski reduces to the city block distance, while $p = 2$ gives the Euclidean distance and $p = \infty$ yields the Chebychev distance.

Cosine:

$$d = 1 - \frac{\mathbf{w}_i\,\mathbf{w}_j^T}{|\mathbf{w}_i|_2|\mathbf{w}_j|_2}$$

Correlation:

$$d = 1 - \frac{(\mathbf{w}_i - \overline{\mathbf{w}}_i)(\mathbf{w}_j - \overline{\mathbf{w}}_j)^T}{|(\mathbf{w}_i - \overline{\mathbf{w}}_i)|_2|(\mathbf{w}_j - \overline{\mathbf{w}}_j)|_2}$$

where

$$\overline{\mathbf{w}}_i = \frac{1}{|V|}\sum_{k=1}^{|V|} w_{ik},\ \overline{\mathbf{w}}_j = \frac{1}{|V|}\sum_{k=1}^{|V|} w_{jk}$$

Jaccard:

$$d = \frac{\#\left[(w_{ik} \neq w_{jk}) \cap ((w_{ik} \neq 0) \cup (w_{jk} \neq 0))\right]}{\#\left[(w_{ik} \neq 0) \cup (w_{jk} \neq 0)\right]}$$

2) Clustering Algorithms: LDA is essentially pulling out the 'principal components' from the text documents, reducing a large archive of data into just K representative topics $\beta_{1:K}$. Therefore, our assumption is that by grouping similar particles induced from past topics, we can observe the clustering patterns and make reasonable predictions on how the future topics will be like.

Now we have a collection of well-defined distance functions, we can start looking at clustering methods to group our particles together accordingly, and here are a set of common clustering algorithms we will consider:

K-Means: As the name itself suggests, the K-Means clustering algorithm consists of K cluster centroids and moves these means iteratively towards the center of its closest neighbors, until they no longer change. Although K-Means has been proved to be guaranteed for convergence [17], its clustering performance is often correlated to how the seeds are initialized at the beginning, and the optimal choice of K is often not apparent (in our case, it is the same as the number of topics.)

```
1. Initialize the means by picking
   K samples at random
2. Iterate
2.a. Assign each instance to
     its nearest mean
2.b. Move the current means to
 the center of all the
 associated points
```

Hierarchical Agglomerative Clustering (HAC): This algorithm starts with treating every instances as individual cluster, and iteratively joins pairs of similar clusters repeatedly until there is only one. If we take the merging history and form a hierarchical binary tree, it will look like the dendrogram in Fig. 5.

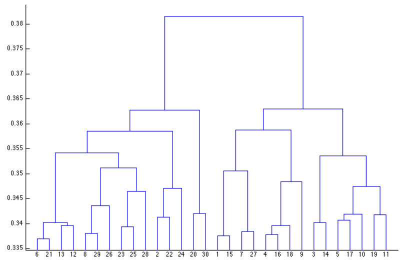


Fig. 5: A sample dendrogram by HAC

Although we have defined the distance functions in the previous section, we have yet formalized the similarity functions between clusters. In our method, we will focus on the following four similarity functions:

- **Single-linkage:** Also known as the nearest neighbor, computes the distance between the two closest elements from two clusters

- **Complete-linkage:** The conjugate of **single-linkage**, also known as the farthest neighbor, computes the distance between the two farthest elements (maximum distance) from two clusters
- **UPGMA:** Unweighted Pair Group Method with Averaging calculates the distance between two clusters, $C_i \& C_j$, by averaging all distances between any pair of objects from the two clusters.

$$dist(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{w_i \in C_i} \sum_{w_j \in C_j} dist(w_i, w_j)$$

Now, let's call this newly-formed cluster C_{ij} and compare its distance with another cluster C_k:

$$dist(C_{ij}, C_k) = \frac{|C_i| dist(c_i, c_k) + |C_j| dist(c_j, c_k)}{|C_i| + |C_j|}$$

where $c_i,\ c_j,\ c_k$ are the centroids for clusters $C_i,\ C_j,\ C_k$

- **WPGMA** Weighted Pair Group Method with Averaging is similar to UPGMA except that the cluster distance is now calculated as:

$$dist(C_{ij}, C_k) = \frac{dist(c_i, c_k) + dist(c_j, c_k)}{2}$$

The behavior of HAC is often dominated by the chosen similarity function. While each variant has a different set of clustering patterns it is good at capturing, all of them have certain vulnerabilities.

3) Putting It Altogether: Now we have gone through all the necessary concepts we used, the intuition behind our Thematic Particle Clustering (TPC) algorithm is pretty straightforward, as shown in Table. II.

Step	Action
1	Initialize $\mathbf{w}_{1:N} = \beta_i^1 + \mathbf{N}(0, \sigma(\beta_i^1))$ // Randomly assign each particle the i-th topic from the first year β_i^1
2	For $y = 2 \cdots (yo - 1)$ // Iterate the following steps through each of the following years in the time sequence until reaching right before the predicting year yo
3	$\mathbf{c} = cluster(\mathbf{w}_{1:N}, K, dist, link)$ // Get the cluster index $\mathbf{c}$ from the clustering algorithm with the designated distance function $dist$ and the linkage $link$
4	For $k = 1 \cdots K$
5	$\mathbf{w}_k = \gamma\, \mathbf{w}_k + \beta_j^y + \mathbf{N}(0, \sigma(\beta_j^y))$ // Iterate through each of the K clusters to find the closest topics from the current year β_j^y, where $\mathbf{w}_k$ is defined as the particles assigned to cluster k. Apply γ, the discount rate, to the old weights and shift the new weights toward β_j^y with a gaussian noise
6	End the for loops for both Step 2 and 4

TABLE II: The TPC Algorithm

IV. Experimental Evaluation

A. Methodology

1) Tree Convergence Graph: The TCG was created out of nessecity to gain intuition about the underlying data structure. TCG helped us formalize our ideas which led to TPC method. TCG very quicky shows when large groups of topics appear. This gave us the intuition that topics could be word biased. Since topics with the most popular words for an year may be more important over time than just an random topics.

add the baseline method (comparing bigram errors)

B. Results

1) Tree Convergence Graph: Here we present an example of a TCG that we used to gain intuition about the topic structure. As you can see below the X axis is the number of years and the Y axis is a listing of 10 different topics each year. New topics (unrelated with past topics in our dataset) show up as unconnected lines to years before. Converging topics all join and begin to follow the same line. In this way we are able to summarize 30 years of data in a single graph.

C. Discussion

1) Tree Convergence Graph: TCGs are a great tool to visualize data but they are not without their tradeoffs. For example a TCG will not directly show you why two topics decided to merge. Nor does it tell you what words a topic is made up of. These issues could be added to the TCG graph but would make the data less parseable. Some other features like word counts for the entire year would just not make sense in the TCG graph. Regardless of some of these drawbacks we still believe visualizing data this way provides an important starting point for building intuition of the data.

V. Related Work

viterbi score

VI. Future Work

Building on this work there are many paths that could lead additional research.

First, our implementation of TPC biases particles with respect to word counts. There could be better features to try and bias results by, for example authors or document length. Finding a better set of features to bias words by could be done by creating an large set of features for each document and seeing which set of features are most correlated from year to year. Correlations could be found through tools like Weka.

Another way to modify TPC's could be through using multi-layer neural network compression techniques on the particles. In the TPC algorithm $N > K$ where N is the number of particles and K is the number of topics. We have to provide a transform to fit N into K so it can be used from year to year. Using a neural network with an appropriately defined error function we could transform N particles into K particles with multiple features weighting the distribution.

soft clustering new features more sophisticated similarity and distance functions

VII. Conclusion

References

[1] Blei, D. Introduction to Probabilistic Topic Models. Princeton University. 2011.

[2] Blei, D., Ng, A. and Jordan, M. Latent Dirichlet allocation. Journal of Machine Learning Research, 3:993-1022, January 2003.

[3] Hoffman, M., Blei, D. and Bach, F. On-line learning for latent Dirichlet allocation. In Neural Information Processing Systems, 2010.

[4] Wallach, H. Topic modeling: Beyond bag of words. In Proceedings of the 23rd International Conference on Machine Learning, 2006.

[5] Griffiths, T., Steyvers, M., Blei, D. and Tenenbaum, J. Integrating topics and syntax. In L. K. Saul, Y. Weiss, and L. Bottou, editors, Advances in Neural Information Processing Systems 17, pages 537-544, Cambridge, MA, 2005. MIT Press.

[6] Blei, D. and Lafferty, J. Dynamic topic models. In International Conference on Machine Learning, pages 113-120, New York, NY, USA, 2006. ACM

[7] Teh, Y., Jordan, M., Beal, M. and Blei, D. Hierarchical Dirichlet process. Journal of the American Statistical Association, 101(476):1566-1581, 2006.

[8] Rosen-Zvi, M., Griffiths, T., Steyvers, M. and Smith, P. The author-topic model for authors and documents. In Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, pages 487-494. AUAI Press, 2004.

[9] Chang, J. and Blei, D. Hierarchical relational models for document networks. Annals of APplied Statistics, 4(1), 2010.

[10] Mimno, D. and McCallum, A. Topic models conditioned on arbitrary features with Dirichlet-multinomial regression. In Uncertainty in Artificial Intelligence, 2008.

[11] Blei, D. and McAuliffe, J. Supervised topic models. In Neural Information Processing Systems, 2007.

[12] Blei, D. and Lafferty, J. A correlated topic model of Science. Annals of Applied Statistics, 1(1):17-35, 2007.

[13] Li, W. and McCallum, A. Pachinko allocation: DAG-structured mixture models of topic correlations. In International Conference on Machine Learning, pages 577-584, 2006.

[14] Reisinger, J., Waters, A. ,Silverthorn, B. and Mooney, R. Spherical topic models. In International Conference on Machine Learning, 2010.

[15] Wang, C. and Blei, D. Decoupling sparsity and smoothness in the discrete hierarchical dirichlet process. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, Advances in Neural Information Processing Systems 22, pages 1982-1989. 2009.

[16] Doyle, G. and Elkan, C. Accounting for burstiness in topic models. In International Conference on Machine Learning, pages 281-288. ACM, 2009.

[17] Selim, S. Z. and Ismail, M. A. (1984). K-means-type algorithms: a generalized convergence theorem and characterization of local optimality. Pattern Analysis and Machine Intelligence, IEEE Transactions on, (1), 81-87.

[18] McCallum, A. K. (2002). Mallet: A machine learning for language toolkit.