Entity Disambiguation with Nonparametric Topic Models

Navin Chandak C. Yeshwanth

Indian Institute of Technology Bombay

November 4, 2016

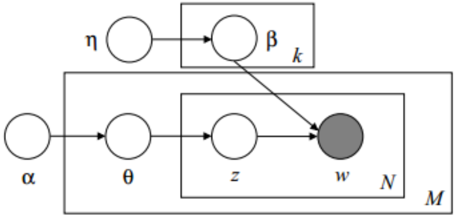
Introduction

- Annotate text with referents in a knowledge base (Wikipedia, YAGO, etc)
- Important preprocessing step for more complex tasks like relationship extraction
- No Attachment entities a central problem of entity linking
- No satisfactory approach to solve the NA problem has been proposed yet
- Most treat NA clustering as a post processing task

Introduction

- Topic models model corpora by treating as a mixture of topics
- Topics are usually distributions over words
- Topic models can be used for entity disambiguation as well
- By extending existing work, we are able to incorporate the NA clustering into the model itself
- Model induces new entities as well as candidate categories for them in the hierarchy

Entity Disambiguation with LDA



- For each entity, sample word distribution $\phi_k \sim Dir(\beta)$
- For each document
  - Sample topic distribution $\theta_m \sim Dir(\alpha)$
  - For each word w_i in document m
    - Sample a topic $z_i \sim Mult(\theta_m)$
    - Sample a word $w_i \sim Mult(\phi_{z_i})$

Consider the Wikipedia annotations A as multinomial observations that bias the distributions of parameters θ_m and ϕ_k

LDA : Making sense

$$P(z_i = k | \vec{z}_{-i}, \vec{w}, \vec{\alpha}, \vec{\beta}, \mathcal{A}) \tag{2}$$
$$\propto \frac{n_{t,-i}^{(k)} + \beta_t + \delta_t^{(k)}}{\sum_{t'=1}^{T} (n_{t',-i}^{(k)} + \beta_{t'} + \delta_{t'}^{(k)})} \cdot \frac{n_{k,-i}^{(m)} + \alpha_k + \delta_k^{(m)}}{\sum_{k'=1}^{K} (n_{k',-i}^{(m)} + \alpha_{k'} + \delta_{k'}^{(m)})}$$

- Word Correlation with window size 1
- Topic Correlation with just same topics
- Similar to global methods of entity disambiguation

Entity Disambiguation with Hierarchical Topic Models

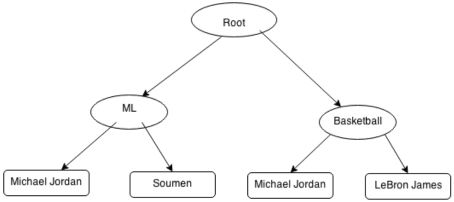
- For each leaf topic k, sample word dist $\phi_k \sim Dir(\beta + \delta_k)$
- For each document m :
  - For each non-leaf topic k, sample topic distribution $\theta_{mk} \sim Dir(\alpha + \delta_k)$
  - For each word w_i in document m:
    - Sample a root-to-leaf path $z_i = z_{i1}, z_{i2}, ..., z_{ili}$ of length li from the topic hierarchy H. For each topic z_{ij} in the topic path, sample child $z_{j(i+1)} \sim Mult(\theta_{m_{zij}})$
    - Sample a word $wi \sim Mult(\phi_{zili})$

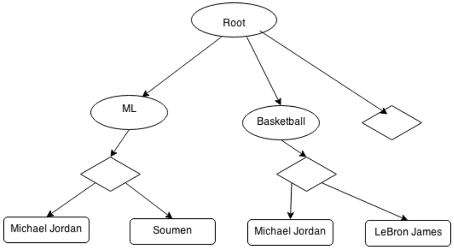
Not Assigned entities : A problem!

- Existing work just identify NA entities
- Wealth of information hidden there : Cluster the NA entities
- Has been said as the last unsolved problem in the field of entity disambiguation
- How do we solve this problem?
- Non-parametric Methods!!

Stepping Aside : Changing the wikipedia category structure

Motivation becomes clear later on





Extending Kataria using CRP

Model

- Infinite number of entities generated using a Chinese Restaurant Process
- While sampling at every special-node, the algorithm either samples an existing entity, or pulls out a new entity with some probability

Problems

- Entities can't have multiple parent
- Addition of same entity multiple times
- Entities at a special-category might not have similar word distributions
- Leaf might not correspond to entities

Extending Kataria using HDP

Model

- Entities are generated globally
- While sampling at a super-category, word can sample an existing entity, or add an entity to the super-category.
- To add the entity, it goes to the franchise wherein the franchise might give an existing entity, or create a new entity

Problems

- New documents will just sample every entity from the same node to ensure topic correlation.
- If not at least everything comes from very close-by nodes.
- Topic Correlation becomes meaningless

Extending Kataria using HDP : Second attempt

Model

- Entities are generated globally
- Super-categories sample a distribution over entities from the global distribution.
- A word goes to the super-category and samples from the distribution over infinite entities

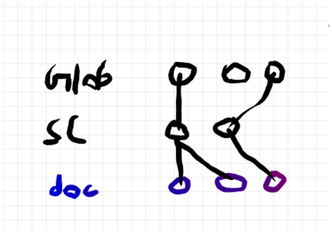
Problems

- Documents will not have independent distribution over entities
- When a document has to go to cricket category, will necessary have high probability of sampling Sachin, because we are sampling from the global distribution

Final Model : Wikipedia Generative Process

- Entities are generated globally
- Super categories sample a distribution over entities
- Documents sample a distibution over entities for every super-category using the super-category distribution over entities as the base distribution
- For each word, go to a special category and generate the distribution over entities from the doc-specific distribution over entities for that super-category

WGP : A View



WGP : Formally I

The parameters to the model are $\alpha, \gamma, \delta, \eta, \zeta$

1. Generate a distribution β over infinite set of entities which will be shared across all the special category nodes

$$\beta \mid \gamma \sim DirichletProcess(\gamma, H_0)$$

2. Sample a word distribution for each of these infinite entities using a dirichlet distribution

$$\phi_k \sim DirichletDist(\eta, \eta, \eta, ...(Wtimes))$$

3. Generate a distribution β_n over the entities in the previous case for each of the special category nodes.

$$\beta_n \mid \beta, \alpha \sim DirichletProcess(\alpha, \beta)$$

WGP : Formally II

4. For each document
    4.1 Sample a distribution over each of the category nodes using a *dirichlet*(δ)

    $$\pi_n \mid \delta \sim \mathit{DirichletDist}(\delta, \delta, \delta ... \mathit{xtimes})$$

    where x is the number of children of the category node

    4.2 Sample a distribution over each of the special category nodes using

    $$\pi_n \mid \zeta, \beta_n \sim \mathit{DirichletProcess}(\zeta, \beta_n)$$

    where β_n was sampled globally for the special category node

WGP : Formally III

4.3 For each word in the document
  4.3.1 Start at the root node.
  4.3.2 Sample a child using
  $$z_t \sim Mult(\pi_n)$$
  4.3.3 Repeat (ii) till you reach an entity/leaf
  4.3.4 Draw the word using the word distribution for that entity
  $$w \sim Mult(\phi_k)$$
  where ϕ_k is the word distribution for the entity chosen in the previous step

Controlling the behaviour of WGP

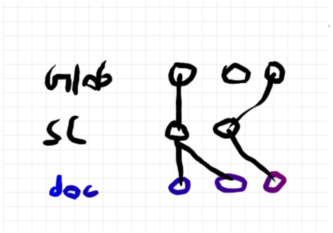
- γ Propensity of creating new entities globally. Set according to type of test documents.
- α Propensity of creating new entities for a special category.
  - We can set this according to empirical knowledge about the super-category (Prime-minister vs Actors). Can learn from data.
  - High because different special categories have different distributions
  - Low for nodes higher up in the tree and high
- ζ Adding new entities for a document. Set according to origin of document.
- η Dirichlet prior for words
- δ Dirichlet prior for distribution over children for category nodes : Control topic correlation

Inference

"What's in a model without a good inference algorithm?" - Great lines said by me

- Show equivalence to Chinese Restaurant Something
- Understand how inference works in Chinese Restaurant Process, and Chinese Restaurant Franchise
- Inference algorithm in Chinese Restaurant Something

Chinese Restaurant Something I



- Each word goes to document, samples a topic from the document
- Sampling a topic from document : Give one of the existing topics with probability proportional to the number of words associated with a particular topic or create a new one for the document(picks it up from the special category) with prob prop to: γ

Chinese Restaurant Something II

- Sampling a topic from special category : Existing topic with prob $\propto$ number of times that topic has been sampled OR created a new topic for the special category(pick it up from global distribution) with prob prop to α
- Sampling a topic from global distribution : Existing topic with prob $\propto$ number of times that topic has been sampled OR create a new topic for the global entities with prob prop to ζ

Chinese Restaurant Something : Proof of equivalence

- Consider that the special-category level distributions are given (G_c)
- Posterior distibution of observation given t G_c is given by

$$P(t_n = t | t_1, t_2, ..t_{n-1}, G_c) \propto n_k \phi_k + \gamma G_c$$

- So three-level DP equivalent to sampling the topic from a document and then doing an CRF while sampling from G_c
- That is what the CRS does : Sampling the topic from a document and when required doing a CRF

Inference in CRP : Gibbs Sampling

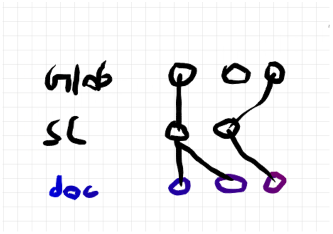
- For each word, sample a table using the posterior probability

$$p(t_n = t | t^{-n}, x) \propto \begin{cases} n_t f_t^{-x_n}(x_n), \text{ if t has been used previously} \\ \gamma p(x_n | t^{-n}), \text{ if t} = t^{new} \end{cases} \tag{1}$$

Inference in HDP

- Variables : Index of table for every word, index of franchise dish for every table
- Infinite set of variables
- Use CRP equation. Use it again
- Sample the dishes on table when required

Inference in WGP



- Use CRP Equation thrice
- Infinite* Infinite set of variables
- Sample indexes on local tables and super-caegory tables when required

Left-over problems : Priors

- Can we enforce a prior on the word distribution based on the category of the entity?
- But every entity has multiple parent categories?
- Which category to choose for enforcing the prior?
- Prior : Do the entities having high topic correlation have similar word distributions?
  - New entities will have high topic correlation with entities it appears with
  - If it appears with some entities, then they are bound to have similar word distributions
  - Pos : "Ganesh Ramakrishnan and Soumen Chakrabarti have worked on machine learning algorithms together"
- Some new entities may still be misclassified
  - Neg : "Sachin Tendulkar and his wife Anjali made an appearance for the event"

Left-over problems : Spurious Topics

- Spurious Topics : Are we creating entities which don't have real world sense. Entities having functional words, for instance?
  - Michael Jordan and LeBron James have played a lot of games together
- Such topics cannot meaningfully and probabilistically be assigned to some categories
- They appear with entities belonging to every category
- Creating new entities for every category will have low probability

Power of the model

- Uses wikipedia effectively for better disambiguation
- Capture word-level correlation of topics.
- Captures topic-level correlation in documents.
- Add new entities to the model with multiple parents at the right place
- It does all the above in a very principled way.

Some auxiliary utilities

- Discover new entities which are not present in Wikipedia in a new document
- Augment the wikipedia category strucuture, by adding new entities at the correct place
- Discover irrelevant category nodes in the wikipedia graph structure ("list of people born in 1952")
- Discover misclassified entity pages in Wikipedia

System Design

- The following questions need to be answered:
  - Which documents do we train on?
  - Which documents should we test on?
  - Which Hierarchy to use?
- Choice of test set motivates other choices made

System Design - Which test set?

- Until recently, NA clustering hasn't received much attention
- Entity Recognition and Linking task at TAC was the first competition to add NA Clustering to the task description
- Evaluates both:
  - The ability to link to known entities
  - Cluster entities across documents
- Metrics which penalize impure clusters or extremely small clusters are used in evaluation
- Precisely the type of test set to run our model on

System Design - Where do we train?

- Unfortunately, the TAC test set does not come with training data
- Need a large annotated set of documents for training
- TACs Knowledge Base is constructed as a subset of Wikipedia
- Training on Wikipedia documents will let our model learn on TAC entities as well
- The vast number of annotations on Wikipedia make it a natural choice for a training set

System Design - Which Hierarchy do we use?

- TACs KB does not have a category hierarchy
- Wikipedia's Category Graph is very rich with diverse categories
- But with a fair number of negatives as well
- The presence of spurious topics
  Example: 20-th Century Births
- No clear rule to define the relatedness of topics
- Leads to cycles and self-loops - Need to prune graph before using it

System Design - Pruning the Wikipedia Category Graph

- First requirement is that graph should have a single root node
- Would like to add a root node which adds the fewest number of edges
- Otherwise topics lower in the hierarchy will be given unnecessarily high probability
- The following algorithm turns out to be optimal
    1. Run an algorithm to compute strongly connected components on the category graph G
    2. Let c_i with $i \in 1 \dots k$ denote the strongly connected components in G with no incoming edges
    3. Connect the root node n_r to one element in each of c_i for $i \in 1 \dots k$

System Design - Pruning the Wikipedia Category Graph

- Next, we will need to ensure that we have a DAG by pruning cycles and self-loops
- Would like to retain edges closer to the root because they affect more documents
- Documents which use far away edges already have significant topic information
- If an edge is to be pruned, we'd prefer it not to move nodes far away
- We've used an algorithm which will preserve *all* shortest paths to a node

System Design - Pruning the Wikipedia Category Graph

1. Let $S = \phi$ to be the set of edges to be retained
2. Start a Breadth First Search from the root n_r
3. For each edge, e encountered during the BFS, check if it forms a cycle with the edges in S
4. If the edge, e does form a cycle, mark it to be deleted
5. Otherwise, add it to S
6. Repeat the above steps till all nodes and edges have been seen

System Design - Can we train?

- TAC has approximately 800,000 entities
- Training set from Wikipedia would have about 2M documents
- Each document has on average close to 400 words
- The category graph has 900,000 nodes
- Assuming, each word is sampled only once, close to $O(10^{20})$ operations
- Need to explore strategies to speed up inference

Speeding Up - Prune Category Graph

- Not many nodes required for modelling topic correlation
- Even as few as 12 nodes leads to good performance
- Can afford to prune the category graph heavily
- With several heuristics, can bring it down to 3400 nodes
- Would like to identify important nodes in the graph
- Nodes should simultaneously cover a lot of documents while not being too similar to each other

Speeding Up - Prune Category Graph

- Borrowed the work of Ramakrishna Bairi whose research focuses on similar problems
- Obtained a set of 1000 nodes which best describe the 800,000 TAC entities
- Problem of extracting subgraph from the 1000 nodes
- We implemented three heuristics to prune the graph:
  1. Remove all nodes which cannot reach any of the Salient Nodes (10.5K Nodes Left)
  2. Remove all the nodes with only a single outgoing edge in the reduced graph (5.5K Nodes Left)
  3. Remove nodes for which there exists a child which can reach the same number of Salient nodes as the parent (3.4K Nodes left)

Speeding Up - Faster Inference Procedures

- Two main heuristics:
  1. Caching Probabilities in the graph
     Does not change the working of the model
  2. Blocking topics for documents based on some easy to compute information
     Changes the working of the model
- The effects of both are complimentary
- Together, we obtain significant speed-up

Speeding Up - Caching Probabilities

- Would like to maintain $P(w|n)$ at each node in the graph G
- G being a DAG provides us with a nice recursive formulation

$$P(w|n) = \sum_{i=1}^{k} P(n_c|n)P(w|n_c)$$

- Now, consider a super category node $P(w|n)$ can be defined as follows:

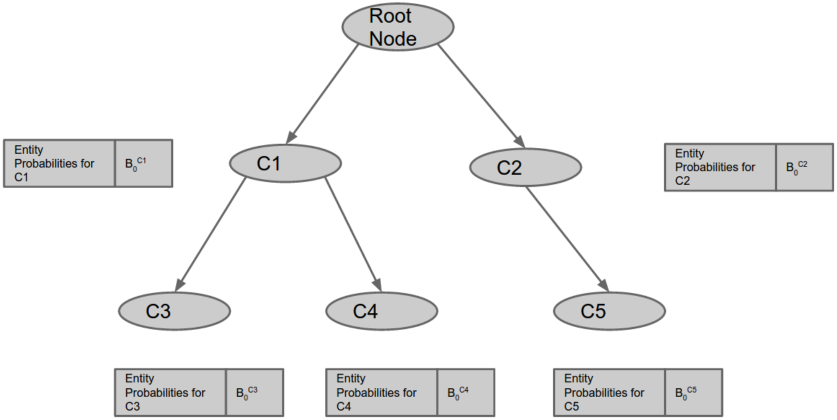
$$P(w|n) = \sum_{i=1}^{k_d} \frac{n_{id}}{\sum_{i=1}^{k_d} n_{id} + \alpha_{nd}} \phi_{t_{id}}(w) + \frac{\alpha_{nd}}{\sum_{i=1}^{k_d} n_{id} + \alpha_{nd}} \int_t \phi_t(w)\beta_n(t)dt$$

$$\int_t \beta_n(t)dt = \sum_{i=1}^{k} \frac{n_i}{\sum_{i=1}^{k} n_i + \alpha_n} \phi_{t_i}(w) + \frac{\alpha_n}{\sum_{i=1}^{k} n_i + \alpha_n} \int_t \phi_t(w)\beta(t)dt$$

Speeding Up - Caching Probabilities

- $P(w|n)$ can be written as a combination of all the entities that are reachable from it and the global distribution over entities
- The coefficients of these terms does not change if the supercategory node sampled is not reachable from n
- How it can affect $P(w|n)$ through the distribution of the entity chosen
- Coefficients only change for the ancestors of the supercategory node picked
- In a tree, at most H nodes will have changed coefficients
- At most one coefficient will be changed at each node along the path chosen

Caching Probabilities



Root Node
C1
C2
C3
C4
C5
Entity Probabilities for C1
B_0^{C1}
Entity Probabilities for C2
B_0^{C2}
Entity Probabilities for C3
B_0^{C3}
Entity Probabilities for C4
B_0^{C4}
Entity Probabilities for C5
B_0^{C5}

Figure: Illustration of Caching Probabilities

Speeding Up - Blocking Entities

- Caching probabilities exploits the structure of the Graph to speed things up
- Blocking techniques help reduce the number of entities to be considered
- Use simple methods to obtain small set of entities for documents
- Only perform inference keeping these topics in mind
- Reduces number of entities considerably
- Can be interpreted as regularization for the model

Quick Evaluation

- Now possible to run on the whole of Wikipedia
- But a run can take a few days
- Need a smaller dataset for quick evaluation
- Clean category graph is preferred
- YAGOs Taxonomy graph is perfect for this

Using YAGO

- Combines Wikipedia's category structure with WordNet's carefully assembled hierarchy
- WordNet has groups of words related by hypernym (subConceptOf) relations
  Example: People is a hypernym of entertainer
- WordNet's hierarchy has clean well defined notions of a relationship between words
- YAGO contains mappings from Wikipedia's categories at the bottom of the category graph to WordNet's hierarchy
  Example: Wikipedia category "Northern Districts cricketers" is linked to Wordnet Entry "cricketer 109977326"
- But, it is not as complete as Wikipedia

Using YAGO

- New dataset restricted to scientists
- WordNet provides very clean subcategories for scientists
  Example: Biologists, Computer Scientists, etc
- Would expect good topic correlation at this level
- Using the WordNet, section of the YAGO hierarchy, we get a training 84 nodes and 5900 entities
- Training documents extracted from Wikipedia having a good number of occurrences of these entities
- We get a moderately sized training set with about 5000 documents

Experimentation - Theoretical Estimates of the Running Time

- Thanks to the optimization which caches probabilities along the nodes of the graph, we get a theoretical running time for inference for interation for a single word to be:

$$\mathcal{O}(num\_words * num\_entities * depthofthetree)$$

- Multiplied by the number of documents and the number of words in a document, we get a theoretical estimate of for one iteration through the training set

$$\mathcal{O}(num\_document * num\_iterations * num\_words * num\_entities * depthofthetree)$$

- With the parameters of the YAGO training set, we get an average estimate of the running time to be 0.6 seconds (with 3.5k entities and 150 words per document on average)
- The product comes out to be 0.05 seconds. Hence there is a constant overhead of 10.

Training and testing framework

- Train only on documents within the reduced set of entities
- A filter on number of occurrences is ineffective
- Longer documents with relatively few mentions of the relevant entities are selected
- A better metric would be the relative occurrences of the relevant entities with respect to the other entities in the document
- A threshold can be used to set the purity of the selected document
- Complete the annotations on the documents for better results
- Amongst the documents selected, every 10th document is used for testing

Results on YAGO Subset

- An overview of the training data
  - Number of entities : 6000 entities
  - Number of nodes in the graph : 83
  - Number of edges in the graph : 84
  - Number of documents : 4953
- But only 3590 entities have any mention at all

Experimental Results - 5k Documents Without Subsetting

- When the number of subsetted documents is 5k, then 3.5 k out of the 6k entities are active
- One iteration over 5k documents 50 minutes
- 725 out of 4139 annotations were correctly marked after 1 iteration
- Due to the high training time, multiple iterations become prohibitive

Experimental Results - 5k Documents With Subsetting

- Single iteration cost goes down to 6 min
- Only 439 out of 4139 entities were marked appropriately
- 725 out of 4139 annotations were correctly marked after 1 iteration
- The ratio of speed up is not so good perhaps because a lot of entities are being created

Experimental Results - 500 Documents Without Subsetting

- Without subsetting
  - Number of subsetted documents is 500, then 1.5 k out of the 6k entities are active
  - The time taken per iteration is 320 seconds
  - It reaches an accuracy of about 200 correct annotations out of 786 annotations.
- With Subsetting
  - 10 seconds to perform a single iteration
  - It reaches an accuracy of about 200 correct annotations out of 786 annotations.
  - Achieve speedups of around 35x

Experimental Results - Effects of Change in Parameters

- Without entity subsetting, 1.5 k entities and 350 words/doc on average takes 0.6 seconds per doc on average
- Similarly 3.5k entities and 140 words on average per document again takes 0.6 seconds on average
- Running time increases linearly with number of entities
- With a single iteration, the number of correct annotations, with 500 documents and 1.5k entities was 174 mentions out of 786
- When we did 2 iterations, then the number rose to 200 out of the 786.

Experimental Results - Effects of Change in Parameters

- Model is extremely sensitive to hyper parameter setting
- Good hyper parameter settings are a must
- When we changed the changed the eta parameter from 0.05 to 0.2, number of correct annotations fell from 200 to 69

General Observations - Sparsity of Training Data

- The training data is extremely sparse
- Most of the entities occur only once and only a few of them occur frequently
- The average number of words per entitiy in the subset of 5000 documents was just 12
- This means that each entity only has 3 or 4 annotations
- About 35% of the entities had just a single annotation out of the 3.5 k entities that were marked in the data set

General Observations - Blocking Techniques

- The spotter used is incomplete in its suggestion of candidate entities
- On manual check on a few documents, they seem to give only 25% to 40% of the candidate entities
- We need a customized code can also provide candidate entities from the new entities that were created
- We plan to implement our own spotter to help with this

Setup for Future Experiments

- The setup for full-blown experiments with the TAC dataset is already setup
- The data has been prepared for this experimentation
- The expected running time assuming the following parameters
  - number of documents $= 1000000$
  - number of words $= 250$ per documents
  - number of entities $= 50$ per document because of entity subsetting
  - depth of tree $= 8$ (YAGO graph)
- Time per iteration will be 10^{11} operations
- Time for training will take 2 days

Structure Learning - Why learn structure?

- Why not learn the hierarchy from the data?
- Some reasons to learn structure
  1. Structure is insufficient or ill suited to the problem at hand
  2. Structure needs to evolve over time
- First reason relevant to our case
- Will modify existing hierarchy learning models

Structure Learning - Nested CRPs and Nested HDPs

- Both nCRPs and nHDPs organize topics as part of an infinitely branching tree
- Documents select distributions over nodes of the tree and generate words
- Goal is to learn general topics near the root of the tree and specific topics as the depth increases
- nCRPs and nHDPs differ in how they choose this distribution over the nodes of the tree

Nested CRPs

- Let τ represent an infinitely branching tree with topics at each node
- Topics are drawn from a Dirichlet distribution
- How does a document select a distribution over nodes?
  - Each document samples a path from the root of the tree τ to a leaf at depth L
  - For each word, selects a node along the path and generates word according to that topic
- The path probabilities remain to be defined

Nested CRPs

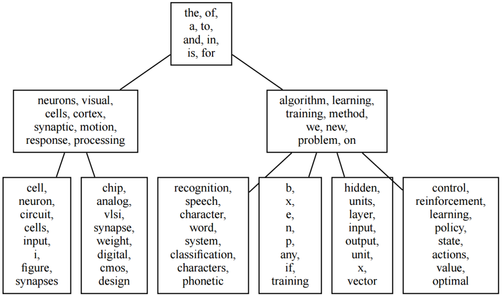
- Definition of path probabilities bring about correlation between documents
- Probability of choosing a path decomposed into conditional probabilities of choosing outgoing edges at each node in the graph
- Let $c_1, \ldots c_L$ denote the nodes on the path to the leaf. c_1 is always the root of the tree. For each $l \in 2 \ldots L$, choose a path as follows:

$$P(c_l = n_j | c_{l-1}) = \frac{n_{c_{l-1}j}}{(\gamma_i + m_{c_{l-1}})} \text{ if } n_j \text{ has been sampled previously}$$

$$P(c_l \text{ is a new node}) = \frac{\alpha_i}{(\gamma_i + m_{c_{l-1}})}$$

Nested CRP Results

- Since, documents sharing the same path share similar topics, one would expect that the more they share a path, the more similar the documents will be
- An empirical evaluation of the nested CRP is shown below on abstracts from the NIPS conference



the, of, a, to, and, in, is, for
neurons, visual, cells, cortex, synaptic, motion, response, processing
algorithm, learning, training, method, we, new, problem, on
cell, neuron, circuit, cells, input, i, figure, synapses
chip, analog, vlsi, synapse, weight, digital, cmos, design
recognition, speech, character, word, system, classification, characters, phonetic
b, x, e, n, p, any, if, training
hidden, units, layer, input, output, unit, x, vector
control, reinforcement, learning, policy, state, actions, value, optimal

Figure: Results of Running the nCRP on NIPS Abstracts

Nested HDPs

- In a nCRP, a document can sample only a single path
- This means a document belonging to two categories should evolve its own path
  Example: If sports and medicine are two topics below the root, a document on medicine in the context of sports should evolve a new path as it doesn't fit in with either category
- In the nHDP, this restriction is lifted where a document samples its own distribution τ_d over the paths of of the tree
- Also removes the restriction that we can proceed only up till depth L
- In nHDPs, each node draws a new distribution over its children characterized by a DP over the global distribution at that node

Nested HDPs

- But the distributions in τ and subsequently in τ_d are distributions over finite paths
- Need to be converted into distributions over nodes
- This is done by generating a beta random variable, B_n at each node, n
- B_n will denote what fraction of the probability mass entering node n stays at node n
- Therefore, the process of generation of a word n goes as follows:
  1. Sample a node as follows, set $n_w = root$
     1.1 With probability B_{n_w}, sample n_w
     1.2 Otherwise, sample $B_{n_w} \sim G^d_{n_w}$ and repeat
  2. Sample a word according to t_{n_w}

Why learn structure for disambiguation?

- One reason is because the category structure might be ill-suited to disambiguation
- Here are some of the top categories for both the YAGO and Wikipedia Category structures

Wikipedia	YAGO2s
Years	artist
Tracking_categories	creator
Wikipedia_maintenance_categories_sorted_by_month	person
Works_by_type_and_year	causal_agent
Land_counties_of_Poland	physical_entity
Populated_places_in_the_United_States_by_county	object

Table: Some Top Categories for Wikipedia and a Path of Subtypes for YAGO

A Structure Learning Model for Disambiguation

- We would like to place priors over the nodes of a tree like the nCRP or the nHDP
- But we do not need to learn topics at different levels of generality
- One key difference, we do not want the ability to pick topics at every level
- We would like to have only nodes away from the root to generate topics
- Also, keeping in mind our model, we want these nodes to generate a distribution over the same topics

A Structure Learning Model for Disambiguation

- Like the nCRP, we will have an infinitely branching tree with finite height L
- The leaves of this tree will be the special category nodes
- Similarly to the WDA model, also presented here, each leaf draws a global distribution over entities and a distribution over entities for each document
- Unlike the nCRP, however, we would not want each document to sample only a single path
- To this end, we will allow each document to draw a distribution τ_d over the paths of the infinitely branching tree
- Each word will then choose a path to a leaf, pick an entity according to its distribution over entities and sample a word

A Structure Learning Model for Disambiguation

- The two models proposed for disambiguation are closer than they appear
- If we replace the infinitely branching tree with a finitely branching tree of height L, the structure learning model reduces to WDA
- This model even though it was proposed for a tree can be easily generalized to include DAGs
- A DAG will have greater power to model correlations because topics with multiple parents like “politics in sports” will have both politics and sports as parents but this isn’t being captured in the tree

Steps Ahead

- Learning hyper-parameters : Cross-validation vs Priors on top of hyper-parameters
- Minka iteration method as described in Kataria
- Gibbs sampling stopping criterion : Current vs others
- Blocking techniques : Ashish's code doesn't peform very well.Need to write our own code
- Using other features : Entity page

Fall back options

- Distributing the inference : Parallel inference if needed. Very good basis to learn parallely on wikipedia. Delayed updates. Have worked on it before
- Structured Learning of the category graph (graphs which are better suited for entity disambiguation)
- Giving more importance to close words (thought of models on LDA)

Difficulties Faced During the Project

- The first hurdle was finding an appropriate test set for our model which is meant to discover new entities
- The second difficulty was figuring out mappings between Wikipedia Entities and TAC Entities
  There was significant difference between the entities from October Wikipedia 2008 and Wikipedia 2012 due to the deletion and modification of articles and page IDs.
- The organizers of the ERL challenge were unwilling to make this mapping available to the participants
- We tried several avenues to get the required version of Wikipedia but were unable to do so
- This led to significant amount of time being spent on trying to get good data for the model
- Delays in procuring the code for graph subsetting