Entity Linking Systems and Modeling of No Attachment Entities

by

Navin Chandak C. Yeshwanth

Undergraduate Thesis Report

A dissertation submitted in partial fulfillment of the
requirements for the degree of
Bachelor of Technology

in

Computer Science and Engineering

under the guidance of
Prof. Ganesh Ramakrishnan and Prof. Soumen Chakrabarti

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Mumbai, India

To our parents

Abstract

The Semantic Web is an ongoing effort to make information found on the web more amenable to manipulation by computers. A document in this framework goes beyond a simple collection of text. This view of the web tends to lose out important information about the real world referrents of the text in the document. The idea of the Semantic Web is that the web is viewed instead as a collection of information about real life entities and the events or relationships that pertain to them. This new view of the web can greatly help enhance user experience both while browsing a webpage as well as aiding search engines in retrieving more relevant documents. A major challenge to the realization of this idea is that the amount of semantic information on webpages is extremely limited in comparison to the size of the web. Therefore, there isn't much impetus to develop applications which exploit this new dimension. The task of automatic entity linking is to increase the quantity of semantic information on webpages by annotating text to their referrents on Wikipedia. While Entity Linking has been extensively researched, its applicability to general web text is still extremely limited. There are several key challenges that still need to be overcome before such a system can be deployed on a large scale. The main purpose of this thesis is to highlight the problems which plague current systems and discuss proposed strategies to tackle them.

Contents

Chapter 1

Introduction

The Semantic Web is an ongoing effort to make information on web pages more amenable to manipulation by computer systems. In this framework, a webpage is viewed as a collection of entities which expresses some information which relates to them. For example, rather than thinking of the sentence,"Narendra Modi is the Prime Minister of India" as a collection of eight tokens, it is thought of as expressing the relationship "Prime Minister of" between two entities, Narendra Modi and India. The adaptation of this idea has been relatively slow partly due to the fact that there isn't much semantic information on the web for applications to exploit and the lack of applications that require this semantic data.

Entity Linking is the task of disambiguating mention phrases on documents to their referrents in a Knowledge Base like Wikipedia. A good Entity Linking system is a vital component of current efforts to realize the dream of a Semantic Web. By enhancing the semantic information on web pages, it can greatly increase the adoption of the idea of the Semantic Web and help create a platform of the development of new technologies to exploit this new wealth of information. Apart from enhancing user experience, it can also help search engines in pulling up more relevant pages to queries.

While Entity Linking has been extensively researched in the past, there are still several key challenges which limit its applicability for general web text. One of the challenges to Entity Linking systems is the presence of Out-of-Database(OODB) or No Attachment(NA) entites which happens when the entity referred to by a mention phrase does not exist in the Knowledge Base used by the system. Other challenges include the scalability of such systems and the scarcity of annotated data available for training such systems.

A general Entitly Linking system consists of three distinct stages, spotting, candidate-generation and disambiguation. Spotting is the task of extracting mention phrases on a document which can potentially refer to entities in the Knowledge Base. Candidate-generation produces a list a can-

didate entities for a mention phrase. The disambiguation phase annotates the mention phrase with an entity from the knowledge base or a NA label indicating that the referent of the phrase is not present in the database. Several different methods have been proposed for each of the three stages but none of them satisfactorily address the issue of NA entities. We discuss possibilities for extending current systems to handle NA entities in a natural fashion by modelling the evolution of new entities in Knowledge Bases like Wikipedia.

1.1 Organization of the Report

The report is organized as follows. We discuss preliminaries required for understanding entity linking systems in Chapter 2. In Chapter 3, we provide an overview of current disambiguation systems and discuss their drawbacks. In Chapter 4, we discuss Topic Models and the Dirichlet distribution and process which are required to understand the extensions proposed. In Chapter 5, we talk about extensions to existing models to handle NA entities and we finally conclude our report in Chapter 6.

Chapter 2

Preliminaries for Entity Linking Systems

In this chapter, we discuss some preliminaries to understand the workings of current entity linking systems. We talk about the 3 stages of an entity linking system and the concept of a catalog along with a discussion of Wikipedia, which is the catalog which we plan to work with. This chapter can be skipped if the reader is familiar with these concepts already.

The goal of an entity linking system to find and annotate mentions of entities in a piece of text with its referent in a Knowledge Base. Traditional entity linking systems proceed in 3 stages[1]

- *Spotting* - This is the process of finding potential mentions of entities on a piece of text. The mention phrase extracted with some contextual information constitute a *spot*
- *Candidate Generation* - Once a spot is extracted, a set of candidate entities needs to be constructed for it
- *Disambiguation* - Finally, a candidate from the set generated by the previous step needs to be chosen for the spot. If no candidate is found suitable, it is labeled NA.

We now proceed to define a Knowledge Base and specifically, describe Wikipedia, which is the Knowledge Base used in our proposed model.

2.1 Catalog

A Knowledge Base or Catalog for entity linking systems intuitively consists of a type hierarchy in the form of a Directed Acyclic Graph, with Subtype-Of edges between nodes and instances of the types described by the type hierarchy constituting the set of entities contained within the catalog. Therefore, a catalog is fully described by:

- A set of type classes C - This is the set of all possible type classes, an entity in this catalog can belong to. For example, cat can be an abstract type-class which contains all the members of the cat family.
- A set of Subtype-Of relationships S - This a set of Subtype-Of relations between two members of C. If c_1 and c_2 are related by a Subtype-Of relationship, then every entity which has type c_1 also has type c_2. A cycle of Subtype-Of relationships are not allowed within this framework.
- A set of instances I - These are the set of all entities in the knowledge base. Each instance, i has associated with it a set of base type classes C_{i} and consequently also their parents in the Subtype-Of graph. Here, we say that each entity i instantiates the set of type classes C_{i}.

This organization of information is extremely useful for building entity linking systems and can give valuable information regarding the nature of entities described in the piece of text.

2.2 Wikipedia

Wikipedia is a free-access online Encyclopedia which allows virtually anyone with an Internet connection to make changes. Wikipedia is an example of a successful, collaborative effort on an extremely large scale in creating a high quality repository of information in an easily accessible fashion. Despite the fact that Wikipedia was not intended to be a knowledge base, its evolution over the years has made it a strong candidate for use, either directly as a catalog or as a resource to augment information contained in existing catalogs. YAGO is one such example of an ontology which extracts information from Wikipedia. The types of pages found in Wikipedia can be used to describe a catalog based on it.

- Article pages - An article page is a page which describes a particular entity. These pages can be taken to be the set of instances for a knowledge base. For example, the wikipedia page, `http://en.wikipedia.org/wiki/Michael_Jordan`, is a description of the famous basketball player.
- Disambiguation pages - These pages are created for helping users search of ambiguous entities, i.e entities which share mention phrases with other entities. For example, the disambiguation page, `http://en.wikipedia.org/wiki/Michael_Jordan_(disambiguation)` can help users navigate to the right Michael Jordan.
- Redirection pages - Due to the open nature of Wikipedia, the same entity may have several different mention phrases and users searching

with one of these mention phrases are automatically redirected to the right page.

- Category pages - Each article on Wikipedia is an instantiation of a set of categories. Category pages are a useful way to organize these categories and can be used as a proxy for the type classes and Subtype-Of relationships.

While it is true that the vast size and comprehensive coverage of Wikipedia have a lot of advantages, there are also negative impacts on consistency as a result of it's free-to-edit system. One such consequence is the presence of cycle in the category graph. This can be hurtful for entity linking systems which exploit the partial ordering imposed by a Directed Acyclic Graph to make predictions. It is not entirely clear what type of pruning strategy leads to the best results.

Chapter 3

Existing Entity Linking Systems

In this chapter, we provide an overview of past entity disambiguation systems and discuss their shortcomings. While there are several different proposals for entity linking, each of them differing in one or more of the three stages described in Chapter 2, entity linking systems can broadly be classified into one of two categories:

- Local Methods - Local methods only make use of information present in the spot and its context. These methods are oblivious to the presence of other spots in the same document. They only use cues like context matches between spots and the description pages of the candidate entities.
- Global Methods - Global methods in addition to using local cues, also try to enforce a coherence between the entities in the same document. The idea behind these methods is that entities occurring in the same document will tend to be coherent topically. For example, a mention of "Michael Jordan" will refer to the basketball player most of the time, but the present of "Latent Dirichlet Allocation" will strongly bias the disambiguator against this choice of entity and prefer the statistician over the athlete.

3.1 Local Methods

In this section, we will discuss three local methods, Semtag[2], Wikify![3] and Taxonomy Kernels[4]. These methods represent preliminary entity linking systems before the arrival of the more complicated global systems and are representative of early attempts to use knowledge bases for linking.

```
Sim(c, v)
    Let b = argmin_{u∈π(v)} {f_u(c)}
    if b = r return 0
    else return 1
```

Figure 3.1: Sim Function

3.1.1 Semtag

Algorithm

Semtag was one of the first papers to perform automatic entity linking. It uses a simple algorithm to perform entity linking. The catalog used to disambiguate entities is the Stanford TAP[5] taxonomy. Unlike the other entity linking systems discussed, Semtag does not disambiguate to or instances of the catalog. It disambiguates to nodes in the taxonomy(class-structure) of the catalog. Semtag assumes a set of labels L(u) which consist of all possible labels for instances of the node, u. The three stages for Semtag are as follows.

- The spotting phase consists of searching input documents for the occurrence of labels corresponding to the nodes in the catalog and extracting a window of words surrounding the label as the *context* of the mention. The label with its context is known as the *spot*.
- The candidate generation phase is done automatically by determining all nodes which contain the label in the context.
- Figures 3.1 and 3.2 illustrate the algorithm followed by Semtag to perform disambiguation given a spot, s. The function $f_u(s)$ computes a measure of similarity between the context of the spot, s and the context of the node which is defined by the cumulation of all the contexts in the training set from its descendants. The *tf-idf* cosine similarity is one such notion of similarity. The function, Sim computes whether the algorithm believes whether node u is relevant for context, c. The function, $\pi(x)$ returns the parents of the node, x. The human given scores, m_u^a and m_u^s determine the confidence/probability with which a label corresponding to node, u is on topic and the confidence with which the algorithm correctly determines whether the label is relevant for node, u. From, figure 3.2, we can see that Semtag will side with the decision of m_u^a if it is more confident than m_u^s and will side with the algorithm, TBD, if m_u^s is more confident than m_u^a.

```
TBD(c, u)
    Let u be the nearest ancestor of v with a measure-
ment.
    if | 0.5 − m_u^a |>| 0.5 − m_u^s |
        if m_u^a > 0.5
            return 1
        else
            return 0
    else
        if m_u^s > 0.5
            return Sim(c, u)
        else
            return 1 - Sim(c, u)
```

Figure 3.2: Taxonomy Based Disambiguation

Drawbacks

There are several drawbacks with the methods proposed in Semtag. Firstly, assigning the scores m_u^a and m_u^s is not very principled and it is not going to be possible to do so in larger catalogs than Semtag. Also, since these two scores make blanket statements about all the nodes that come below them, it becomes exceedingly hard to set scores which agree with all the instances that form the descendants of that node. Secondly, Semtag relies only on contexts found previously for particular nodes. It may not always be possible to find such contexts especially given the skew of data found on the web where tail entities are extremely rare and it's hard to find contexts which describe them without extensively searching it. Thirdly, it does not adequately model the presence of entities out of the knowledge base and cannot handle does not give any insight into the behavior of these new entities. Finally, it does not even try to model the case where the same instance can have multiple synonymous names.

3.1.2 Wikify!

Algorithm

Wikify! is another local method which sought to exploit Wikipedia, not for its category structure but rather for the presence of detailed text describing each entity and the presence of anchor text to help disambiguate mentions in web pages. The Knowledge Base in Wikify! is Wikipedia but it makes no attempt to use its category structure to help guide the disambiguation process. Its novelty lies in exploiting data in Wikipedia to generate plausible mention phrases or key phrases in the lexicon of the paper. It uses anchor

text found in Wikipedia itself to come up with a set of possible candidate mention phrases for a particular entity. It then continues to implement two disambiguation algorithms which use features which are very similar to the local features found in the current state of the art. The three stages of Wikify are as follows:

- Key phrase extraction - This is identical to the spotting step described earlier. Anchor text from the different Wikipedia articles is collected and are used as candidates for extracting possible mention phrases from the document. Once the key phrases from the document are extracted, they are sorted and only the top k of them are selected for the disambiguation process. The ranking of key phrases happens through a *keyphraseness* score which is defined as the number of times the phrase, p occurs as a keyword($Count(D_p^{key})$) and the number of times it appears in the corpus, $Count(D_p^{tot})$

$$Keyphraseness(p) \approx \frac{Count(D_p^{key})}{Count(D_p^{tot})} \tag{3.1}$$

- Candidate Generation - This stage is done by virtue of the method in which the candidate mention phrases were generated. Since, the candidate mention phrases were anchor texts for some articles themselves, the set of candidate entities for a mention phrase is the set of entities that mention phrase acted as an anchor for.

- Disambiguation - The disambiguation in Wikify! uses two cues, one from the description page of the entities and the other from the contexts of training data which points to those entities. A notion of similarity like *tf-idf* cosine similarity between the description page of the entity and the context is computed and can be used to make a disambiguation choice. The other method uses words from the contexts of mention phrases that point to that entity. Features are extracted from the set of contexts known to disambiguate to the entity in question and a classifier is used based on how well the features of the new context match the features extracted from the training data.

Drawbacks

Wikify! while resolving some of the drawbacks of Semtag, still suffers from a few problems. Firstly, it only considers disambiguations within Wikipedia. It, therefore, completely ignores the NA problem which is an important part of Entity Linking systems today. Secondly, it does not utilize the category structure of Wikipedia to make predictions. The category structure can have valuable cues to decide which can help make disambiguation easier. And lastly, like all local methods, Wikify! does not impose any coherence in

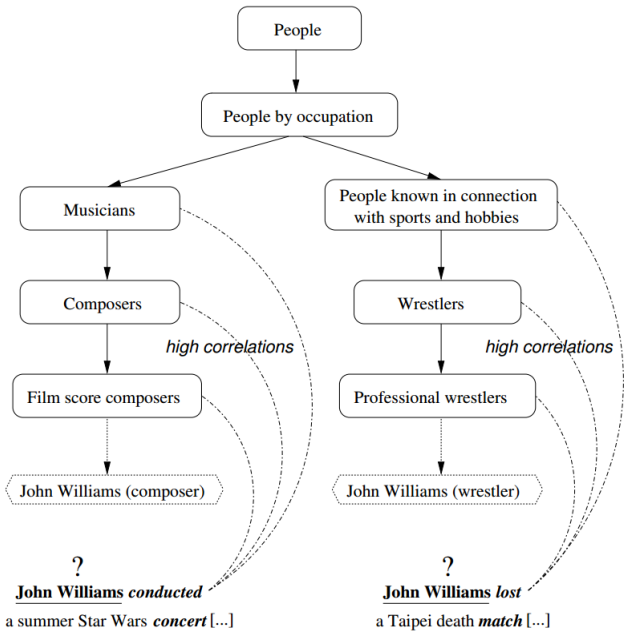


Figure 3.3: Taxonomy Kernels

the entities being discussed in the document. Despite all these flaws, Wikify! sets an important benchmark for entity linking systems and sets a reference point moving forward as several other proposals can be seen as extensions to Wikify!.

3.1.3 Taxonomy Kernels

Algorithm

Taxonomy Kernels were introduced in [4] and were novel in the sense that they exploited the category structure of Wikipedia. By leveraging similarity across all levels of the category structure, they were able to get improved performance than by classifiers like Wikify! which did not use it at all. The key stages in the execution of this system are as follows:

- Spotting - The title pages of all the articles in Wikipedia are searched and a heuristic is applied to extract all titles which correspond to named entities, for example, capitalization cues, etc. For these named entities, their titles, entries from their redirection and disambiguation pages are all added as possible mention phrases to the set of allowed mention phrases. During the testing phase, a document is searched for the presence of any of these mention phrases and the context is extracted to form the spot.

- Candidate Generation - Similar to Wikify!, the set of phrases automatically fixes the set of candidate entities by simply considering the set of all entities that have the mention phrase as a title, redirect or a disambiguation.

- Disambiguation - The novelty of this approach lies in the disambiguation phase. In contrast to other approaches which simply compare the context to the entity description pages, this method compares a context to all the parents of the entity page in the hierarchy. Disambiguation is made by a classifier of the following form.

$$\hat{e} = \arg\max_{e_k} \mathbf{w}\ \mathbf{\Phi}(q, e_k)$$

$$\phi_{cos}(q, e_k) = cos(q.T, e_k.T)$$

$$\phi_{w,c}(q, e_k) = \begin{cases} 1 & \text{if } w \in q.T \text{ and } c \in e_k.C \\ 0 & \text{otherwise} \end{cases}$$

  Here, q stands for the spot that is to be disambiguated, $q.T$ indicates all the terms that occur in the spot, $\mathbf{\Phi}$ is an $|V| \times |C|$ sized feature vector where $|V|$ is the size of the vocabulary and $|C|$ is the number of categories and $e_k.C$ refers to the categories valid for an entity, e_k. The vector, $\mathbf{w}$ is learnt from data extracted using occurrences of the mention phrases as anchor text. This formulation of the feature vector allows Taxonomy Kernels the ability to learn dependencies all the way along the category ancestors for an entity rather than simply using scores computed with entity pages. Figure 3.3 provides an intuitive understanding of Taxonomy Kernels with a score for an $entity, context$ is dependent on all of its parents and not just its own description or sample contexts. It provides an example of how Taxonomy Kernels can be used to distinguish between two possible candidate entities for the mention phrase, "John Williams", the composer and the wrestler.

Despite all of these benefits, this model is still limited because it does not enforce coherency between the entities chosen by the disambiguator. Like all other local models, these also suffer from a poor modeling of entities out of the knowledge base (a simple thresholding of the objective function).

3.2 Global Methods

Global methods differ from local methods in that they try to enforce coherency between the entities disambiguated to on the same document. Spotting techniques employed in these methods are not significantly different from what has been discussed earlier and we will focus only on the disambiguation component in this section. We will look at some disambiguation

techniques from Cucerzan et al[6], which was one of the first papers to consider a coherency factor with the local objective, Cucerzan et al[7], Ratinov et al[8], Kulkarni et al[9] and Han et al[10] whose methods mainly differ in their formulation of the global objective and how they solve it.

3.2.1 Cucerzan

Like Bunescu et al[4], Cucerzan also use the category structure in Wikipedia to improve the accuracy of their disambiguation system. How they differ, is the way they use the category structure. In this setting, the category structure is used to define a coherency score between the entities that a page contains rather than to compute a more robust score of similarity between the context and the entity currently being considered. To properly define the notion of coherence adopted, we need to define some notation:

- C - Set of known contexts from Wikipedia, assumed to be of size M.
- T - Set of categories in Wikipedia, of size N.
- δ_e - A vector of size $M + N$ associated with each entitiy, e, which can be decomposed into two parts
  - $\delta_e|_C$ of size M defined as

$$\delta_e^i|_C = \begin{cases} 1 & \text{if } c_i \text{ is a valid context(mention phrase) for entity } e \\ 0 & \text{otherwise} \end{cases}$$

  - $\delta_e|_T$ of size N defined as

$$\delta_e^i|_T = \begin{cases} 1 & \text{if } t_i \text{ is a valid category for entity } e \\ 0 & \text{otherwise} \end{cases}$$

- d - A vector of size M which stores how many occurrences of each type of context occur in the document
- $\varepsilon(s)$ - The set of entities valid for a context s
- S - The set of contexts in the document with s_i representing the i^{th} context
- e_i - The entity chosen for context s_i in the document
- $\bar{d}$ - A vector of size $M + N$ which consists of d as its first M elements and the remaining N elements are defined as, $\sum_{s \in S(D)} \sum_{e \in \varepsilon(s)} \delta_e|_T$

Now, the objective function maximized by Cucerzan is,

$$\underset{(e_1,e_2,..,e_n)\in\varepsilon(s_1)\times..\times\varepsilon(s_n)}{\arg\max} = \sum_{i=1}^{n} < \delta_{e_i}, \bar{d} - \delta_{e_i}|_T >$$

Coherency in this model is enforced by the $\bar{d}$ term. While this model does attempt to enforce global conditions, the vector $\bar{d}$ is contaminated because it contains terms which correspond to all the entities possible for all the mention phrases in the document. This can hurt the performance of the disambiguator as it introduces noise into the system by including entities which may not in any way represent the contents of the document. The other problem with this method is that it still doesn't address the problem of entities outside of Wikipedia, non-recallables in the terminology of the paper.

3.2.2 Milne and Witten

Milne et al avoid the problem of noise that plagued the previous model. This is due to the way they incorporate global information. Here, coherency within the document is enforced only with respect to a few unambiguous entities in the document. The algorithm proceeds in three stages:

- First, a set of unambiguous mentions in the document are isolated
- The second stage involves computing for all other mention phrases in the document, their relatedness scores with the unambiguous entities. This relatedness score is computed between the unambiguous entity and each of the candidate entities for that particular mention phrase. The relatedness score between two entities a and b is given by the following formula. The sets, A and B are the pages that contain links to the description pages of a and b respectively while W is the set of all pages in Wikipedia.

$$Relatedness(a,b) = \frac{\log(|A \cap B|) - \log(\max(|A|,|B|))}{\log(|W|) - \log(\min(|A|,|B|))} \tag{3.2}$$

- Each unambiguous mention phrase is then weighted in proportion to it's relatedness to other unambiguous mention phrases through the use of the relatedness score and the keyphraseness (from Wikify!) of the particular mention phrase
- These weights are used to obtain relatedness scores for the ambiguous mention phrases by averaging over the relatedness scores with each of the unambiguous entities weighted by the weights from the previous stage

- Finally, a choice has to be made as to how much importance is to be given to the local component and how much to weight the relatedness scores. This is also computed, from the relatedness scores between the unambiguous entities.
- Based on the value obtained in the previous stage, a disambiguation is made for the ambiguous mention phrase weighing either the local or the relatedness cue over the other

While Milne et al might be argued to be better than Cucerzan et al, explained previously, it still has the drawback of being reliant on the presence of these unambiguous mention phrases in the document. It is not guaranteed that these unambiguous mention phrases will be present in the document at all. Also, it does not model the presence of entities outside the knowledge base which make it unsuitable for deployment on general web scale data.

3.2.3 CSAW-2009

CSAW-2009[9] generalizes the methods presented in Milne et al and Cucerzan et al. Unlike Cucerzan, it does not include the presence of entities that the evaluator does not link to in the global objective, thus eliminating the noise from the presence of these entities and unlike Milne et al, it is not reliant on the presence on unambiguous entities and is therefore more robust as it can work in cases where these entities are not available.

CSAW-2009 is also the first paper to incorporate NA entities into the objective function which is a sum of a local and a global compatibility term and is defined as follows.

$$\max_{y} \frac{1}{|S_0|} \left(\sum_{s \in N_0} \rho_{NA} + \sum_{s \in A_0} w^T f_s(y_s) \right) + \frac{1}{\binom{S_0}{2}} \sum_{s \neq s' \in A_0} r(y_s, y_{s'}) \qquad (3.3)$$

- y - The assignments of entities to the set of spots S_0
- A_0 - The set of spots which have been labeled with entities in the knowledge base
- N_0 - The set of spots which have been labeled with NA

In this formulation, the value of ρ_{NA} is the reward for labeling a spot with NA instead of an entity in the knowledge base. The second term is the local compatibility score between the spot, s and the entity it has been annotated to. This is similar to scores obtained in the local methods, except that a mixture of features are used along with a classifier to determine the score of the local comparability. The term on the right, computes the relatedness measures between entities that have been annotated to Wikipedia.

By changing the value of ρ_{NA}, the probability of a random spot being annotated to NA can be manipulated and provides a trade off between aggressively linking to Wikipedia or a more careful approach where only the most sure candidates are linked. While this formulation avoids a lot of the problems of previous global approaches, it still does not adequately model NA entities relegating all of them to a single score. In this formulation, NA entities are completely unrelated to any other entity in Wikpedia which is not true of general web text and also, the local comparability of a spot with an NA entity is considered constant which again does not represent a realistic view of NA entities.

3.2.4 Ratinov et al

Ratinov et al[8] propose a two stage process for entity disambiguation. The first two stages, spotting and candidate generation are not handled within the framework. The difference between the model proposed here and other global models is that the global features are learnt from the results of the local features rather than them being fixed functions. The two stages proposed are:

- Ranker - The Ranker uses a strictly local approach to link a set of mentions, M to the best candidate entity using only local features which are similar to those discussed before, ((td-idf) cosine matches between the text of the document and the context of the mention with the text in the description pages and known contexts for that entity along with prior probability values for that mention phrase)
- Linker - The Linker then proceeds to either assign an NA value to a mention or let it retain the entity given to it by the Ranker based on a global score which is learnt (the features which are fed to the Linker are similar to the relatedness features used by Milne et al except that these features are computed with both in-links and out-links rather than just for in-links). The outputs of the ranker are fed as training data to the linker with their labels being positive if they match the gold standard. This will hopefully guide the Linker to figure out cases where the Ranker can err.

Ratinov et al's approach can be thought of an improved version of both Cucerzan et al and Milne et al. All of these approaches fix the global disambiguation features before the global coherency features are actually used. But Cucerzan et al's global coherency features are noise and are susceptible to include influence from entities which are not even related to the document while Milne et al suffers from a problem on the other extreme where it may not have enough unambiguous entities to make a good judgment. Ratinov et al alleviate this problem by using an empirically tested and stable global

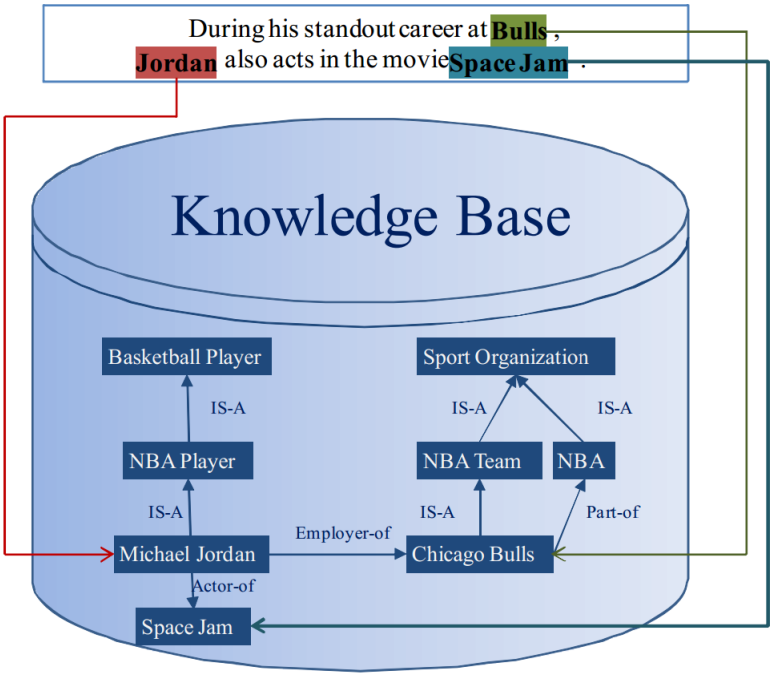


Figure 3.4: Entity Linking in Han et al

disambiguation features(the output of the Ranker). The fact that it learns a classifier over global features also helps improve it's performance in contrast to CSAW-2009 which uses static relatedness features.

Despite these improvements, it is still clear that Ratinov does not adequately model NA entities. This can be seen from the fact that when a mention phrase is given the NA label, it is assumed that that mention phrase has zero relationship with the other entities linked to in the document. This is again not true in general web text. Ratinov et al also suffers from the fact that a wrong label in the Linker is impossible to fix by the Ranker as it does not look at alternative entities for the mention phrase again.

3.2.5 Han et al

Han et al[10] propose an interesting method which tries to model global dependencies beyond the pairwise ones explored by the models seen so far. For example, in Figure 3.4, the correct disambiguations for "Bulls" and "Space Jam" are not related in any way. But the presence of the third entity, "Michael Jordan" in the sentence is enough to relate them. The paper seeks to model these kinds of relationships which are not adequately modeled by pairwise measures as the relatedness of two entities is static in the other models given the model parameters. The disambiguation procedure proceeds

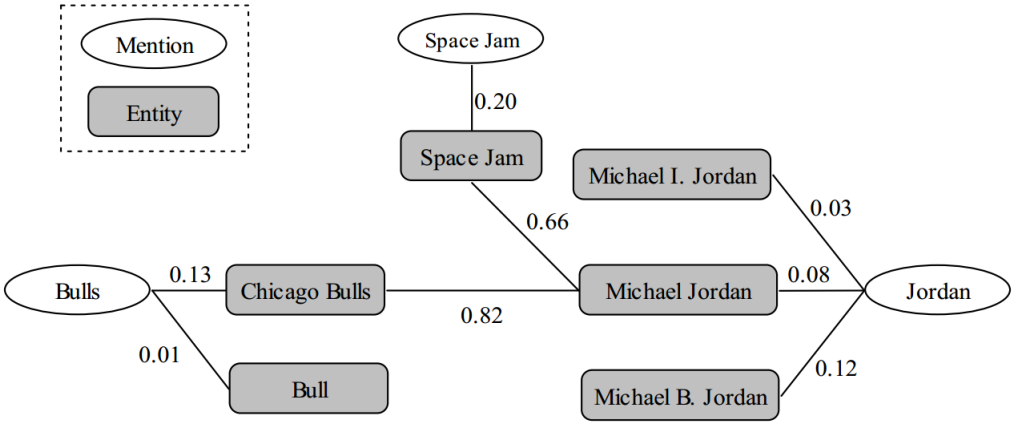


Figure 3.5: Referent Graph Construction

along the following lines:

- First, a referent graph is constructed for all the mentions and their possible referent entities. An example of such a graph is shown in Figure 3.5. The steps for constructing such a referent graph are:
  - Nodes are created for mention phrases and all of their candidate entries
  - Then edges are inserted from the mention phrases to their possible disambiguations with the weights being proportional to the local compatibility score between the mention phrase and the entity
  - Edges are inserted between entities with the weight from e_1 to e_2 proportional to the semantic relatedness between them (this is the same as the measure of relatedness computed by Milne et al), the proportionality constant is chosen such that the sum of the outgoing weights from e_1 is 1
  - Each mention node is given some initial importance proportional to the sum of tf-idf scores that make up the spot for that mention phrase
  - A random walk is simulated along these edges as the outgoing weights are normalized to probability values
  - This is identical to the process for PageRank as well and an analytic solution for the stationary state can be obtained which is equal to $\lambda(I - cT)^{-1}s$ where λ is a parameter which sets the weightage for the restart vector, c is $1 - \lambda$ and T is the propagation matrix for the initial process without the restart

Through this model, multi-node relationships can actually be given sufficient weight as influence can travel from one node to a node situated much farther away. This gives it an advantage over the other global models discussed earlier. Despite this seemingly greater modeling power, it still does not model NA entities. This absence limits the performance of the system due to the large net occurrences of entities outside the knowledge base.

We have seen several models for entity linking and disambiguation and have found that all of them fail to address the problem of NA entities adequately. The importance of these entities is noted by Ratinov et al[8] and is touted as one of the remaining challenges in the design of entity linking systems. The heavy tail distribution of these entities makes the addressing of this problem all the more important to address if significant progress is to be made in entity disambiguation systems.

Chapter 4

Topic Models and Related Concepts

In the following chapter, we cover a few topic models and the concepts underlying their construction which will set up the background required for understanding the later sections of the report.

Topic Models are statistical methods which attempt to explain the behavior of a corpus of documents through a set of topics. The goal is to describe the documents in the corpus as coming from a set of topics which is much smaller than the size of the actual corpus. This is meant to model our intuitive understanding of documents as belonging to one topic or the other and this is mostly determined by the set of words that occur in the document. Therefore, in this setting, a *topic* is a distribution over a set of words, V which will form the vocabulary of our corpus. Before we proceed further, we need to understand the underlying assumptions behind topic models to give insight into why they are constructed the way they are. To do that, we need the concept of Exchangeability.

4.1 Exchangeable Sequences of Random Variables

An exchangeable sequence of random variables is intuitively one that does not care about the order in which these random variables are realized; all possible permutations are equally likely. This is an assumption that is fundamental to the definition of topic models. Topic models are generally assumed to be exchangeable in two respects. One is at the document level, the documents are assumed to be exchangeable and the other is at the word level, the word distribution in a document is assumed to be exchangeable. We will now formally define exchangeable distributions.

Definition

A sequence of random variables, $X_1, X_2...$ is said to be *exchangeable* is for any finite permutation(a finite number of random variables are shuffles between themselves) π, the joint distribution over the permuted sequence is the same as the joint distribution over the original sequence.[11]

$$(X_1, X_2, X_3, ...) \stackrel{D}{=} (X_{\pi(1)}, X_{\pi(2)}, X_{\pi(3)}, ...)$$

It is easy to see from the definition that any sequence of iid random variables is an exchangeable sequence. But, a sequence of iid random variables to explain the documents in a corpus or the set of words in a document would not take into account the relationship between them. Therefore, we need a much richer class of distributions which the concept of exchangeability will provide. We will link the structure of topic models with this assumption of exchangeability through the use of the famous de Finetti's Theorem.

de Finetti's Theorem

Any sequence of exchangeable random variables is a mixture of iid sequences. That is, if $(X_1, X_2, ...)$ are a sequence of exchangeable random variables, there exists a random variable Y on some space S, such that $(X_1, X_2, ...)$ are independent given Y.

de Finetti's theorem now gives us some intuition about how to go about constructing a topic model. Based on the choice of Y, we can define different types of topic models with varying degrees of generalization. The first topic model that we will cover is the Finite Mixture Model.

4.2 Parametric Topic Models

The class of parametric statistics involves placing strong restrictions on the models that we fit to enable better inference. These restrictions usually take the form of enforcing a finite set of real valued parameters[12]. For example, a Gaussian distribution can be fully described by specifying its mean and variance. While this places restrictions on the type of data that one can handle, it can help ease computational costs. Bayesian parametric models assume that the uncertainty in the parameters can be captured through a probability distribution over the parameter space. The parameters which are used to define this distribution over parameters/latent variables are called hyperparameters. In this section, we will cover a few parametric topic models before moving on to nonprametric models.

Finite Mixture Model

In the Finite Mixture Model, we assume that the whole corpus is generated by using a set of K topics and that each document is generated by a single topic. In this model, we have a distribution over words, β_i for each topic, t_i, a prior distribution over the topics which is usually a parameterized distribution by θ, which we will denote by α. We are now ready to describe the generative process for finite mixture models.

- Each document, D, chooses a topic, z with probability defined by α. $z \sim \alpha$
- Each word, w_i in the document D is then generated independently according to the distribution, β_z. $w_i \sim \beta_z$

The finite mixture model is one of the simplest topic models but it does not adequately model large collections of documents where each document has more than one topic. Therefore, we need a more intricate topic model to allow for the possibility of each document having more than one topic.

Latent Dirichlet Allocation

Latent Dirichlet Allocation[13] tries to solve the problems faced by finite mixture models by allowing a document to have more than one topic. One simple method to fix this problem is to simply assign to each document the same distribution over topics. This is not a good solution as documents usually vary in topic mixture components and it is best for each document to have its own distribution over topics. To do this, LDA introduces a prior over the topic space. This needs to be a prior over topic *distributions* and not over topics as in the case of the Finite Mixture Model. The distribution that was chosen for LDA is the Dirichlet Distribution.

Dirichlet Distribution

The Dirichlet distribution, parameterized by a vector α, is a distribution over finite dimensional probability vectors of size n, which is also the length of α. It is defined by the following density function:

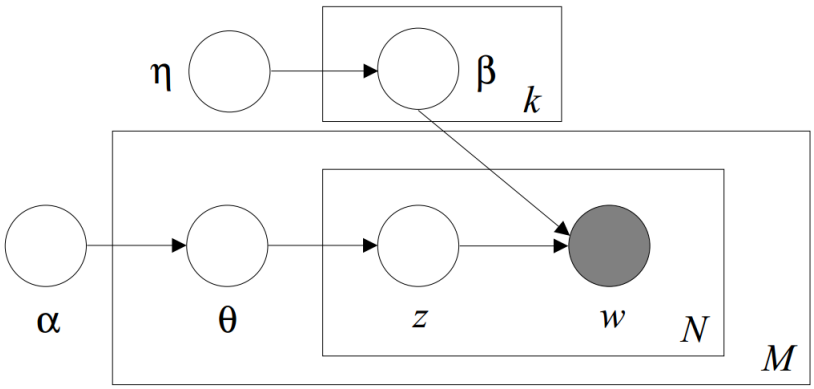


Figure 4.1: Plate Model of Latent Dirichlet Allocation

$$p(x_1, x_2, x_3, ..., x_n) = \frac{1}{B(\alpha)} \prod_{i=1}^{n} x_i$$

$$B(\alpha) = \frac{\prod_{i=1}^{n} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{n} \alpha_i)}$$

$$x_i \geq 0$$

$$\sum_{i=1}^{n} x_i = 1$$

We are now ready to describe the generative process of Latent Dirichlet Allocation which is shown in the convenient plate model convention in reffig:PlateModelLDA.

- Each topic t_i, generates a distribution over words β_i distributed as $Dir(\eta)$
- Each document d generates a distribution over topics θ distributed as $Dir(\alpha)$
- Each word w in document d picks a topic z distributed according to θ
- The word w is then chosen according to β_z

While the Dirichlet model improves significantly upon Finite Mixture Models, it still has some drawbacks. The number of topics is required to be fixed before it can be used. This is hard especially for large collections

of data where we might not explicitly know anything about the content distribution of the data. We would ideally like our model to introduce new topics as and when they are required if we find that the current set of topics do not capture the data well. That is the purpose of Non Parametric Models which we will describe in the next section. The other drawback is that LDA does not model topic correlations where the occurrence of one topic makes another more likely.

4.3 Non Parametric Topic Models

As we saw in the previous section, parametric models usually place pretty strong restrictions on the type of models that one can fit. Also, there seems to be a direct connection between the number of parameters and the complexity of the model that one can fit. By allowing the model to be flexible with the number of parameters, it seems like we can achieve better performance by allowing the algorithm to choose how many parameters it can use. Non parametric models allow data to potentially use an infinitely sized parameter. It must be noted that non parametric are not models without any parameters, rather they allow the algorithm to choose the complexity of the model fit. In this sense, it can be seen as having an infinite-dimensional parameter space. An example of a non parametric method would be the use of splines for regression with a penalty term for model complexity. Bayesian non parametric models analogous to Bayesian parametric models use a distribution over the parameter space and through this framework, the model can be tuned to avoid over fitting. The difficulty in this model is the specification of the prior distribution since unlike the parametric case, is a distribution over a possibly infinite sized parameter space. An analogue to topic models we have considered would be to allow the Dirichlet prior over topics to define a distribution over a possibly infinite set of topics.

Dirichlet Process

As discussed previously, we need a distribution over the parameter space where the parameter itself is infinitely dimensional. One such distribution is the Dirichlet process. We will define the Dirichlet process now.

Let H be a distribution over Θ, which is a probability space. We say that a random distribution, G over Θ is Dirichlet Process distributed with respect to H and a positive real number α if for any finite measurable partition of Θ, $(A_1, ..., A_r)$, $(G(A_1), ..., G(A_r))$ is distributed according to $Dirichlet(H(A_1), ..., H(A_r))$.[14]

$$(G(A_1), ..., G(A_r)) \sim Dirichlet(H(A_1), ..., H(A_r))$$

We are now in a position to start talking about generalizations of the

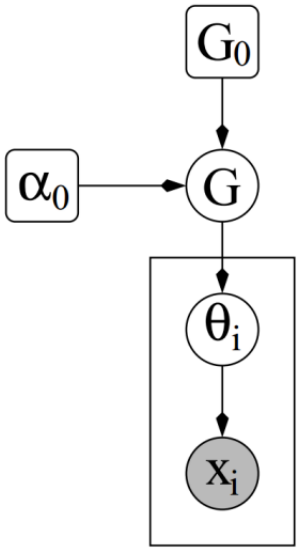


Figure 4.2: Plate Model of the Dirichlet Process Mixture Model

models discussed previously. We will describe the Dirichlet Process Mixture Model and the Hierarchical Dirichlet Process which are generalizations of the Finite Mixture Model and the Latent Dirichlet Allocation from the previous sections.

Dirichlet Process Mixture Models

In the Dirichlet Process mixture model, each document picks a topic and generates each word according to that topic similar to a Finite Mixture Model. The difference in this model is that the set of topics under consideration is potentially infinite while the Finite Mixture Model was limited to a fixed set of K topics. The plate diagram for the Dirichlet Process Mixture Model is shown in 4.2. The base distribution, G_0 used is generally a Dirichlet distribution as we want a distribution over topics and the Dirichlet Process provides us with a distribution with the same support as the base distribution which in this case is the space of all possible topics. A generative description of the process is as follows:

- First, G is sampled according to the Dirichlet Process with G_0 as the base distribution and α_0 as the concentration parameter. $G \sim DP(\alpha_0, G_0)$
- Each document then samples a topic θ from G. G

- For each word w is then generated according to the multinomial distribution represented by θ. $w \sim$

To provide some intuition, G can be seen as the global pool of topics that is generated by the Dirichlet process along with their probabilities. This is analogous to the set of topic vectors in the Finite Mixture Model except that here, we have an infinite set of topic vectors rather than the finite that we had in the Finite Mixture Model. Each document then proceeds to pick a topic from the global pool, G, and then proceeds to generate words based on the distribution of this topic. Some important questions have to be addressed before we continue. If each document can select from a global pool of possibly infinite topics, it seems a cause for concern as there can be a tremendous amount of overfitting. This was the reason we did not directly sample from G_0 for each document to begin with. So, why bother with Dirichlet Process Mixture Models in the first place. The second problem, a problem related to the first, is that we may never have two documents picking the same topic given the intuitive understanding that the set of possible topics is extremely large. So, how can two documents be compared if they pick two different topics all the time. We shall see that while the global pool of topics, G, is quite large, infinite in fact, it is discrete and the tendency for two documents picking the same topic is in fact quite high. This will be seen through the posterior distributions for G.

Posterior Distribution of G

An intuitive understanding can be seen through the posterior distribution over G. Consider the case where $\theta_1, ..., \theta_n$ have been drawn from G and now we wish to find the posterior distribution over G. We will not repeat the proof here but it can be proved quite simply from the definition of Dirichlet processes[14]. The posterior distribution over G also happens to be a Dirichlet process with a modified concentration parameter and base distribution and notably, one that is discrete at the points where draws have been observed and moreover the proportion of the base distribution in the posterior keeps decreasing with each new draw from the base distribution.

$$G|\theta_1, ..., \theta_n \sim DP(\alpha + n, \frac{\alpha}{\alpha + n} G_0 + \frac{n}{n + \alpha} \frac{\sum_{i=1}^{n} \delta_{\theta_i}}{n})$$

where δ_θ denotes the Kronecker delta function at θ

Further Draws from G

Now, we will look at the distribution of θ_{n+1} given that $\theta_1, ..., \theta_n$ have already been drawn from G. Coincidentally, this also happens to be the same as the posterior base distribution of G.

$$\theta_{n+1}|\theta_1, ..., \theta_n \sim \frac{\alpha}{\alpha + n} G_0 + \frac{n}{\alpha + n} \frac{\sum_{i=1}^{n} \delta_{\theta_i}}{n}$$

This assigns a significant amount of probability to the possibility that θ_{n+1} is the same as one of the θs seen before and this probability keeps increasing with more and more draws from G_0. This happens because G is discrete and moreover, it discourages the drawing of new θs as more and more are drawn. This reluctance to draw more and more new topics in the context of topic models is what makes it ideal to the topic modeling setting as we want as few topics as possible but which describe the data well.

The Chinese Restaurant Metaphor

Now, well present an alternative picture of the process of generation of s from a sequential perspective. The metaphor involves a group of customers walking into a restaurant and setting down at tables. Each table represents a value in support of the base distribution and the customers at that table are the ones who have chosen that parameter in the support of the base distribution. Let the customers be $(c1, ..., cn)$ entering the restaurant in that order and so the metaphor goes:

- The first customer c_1 picks a value of θ and settles down at that table
- Each following customer sits at a table with probability proportional to the number of people at that table and sets up a new table with probability proportional to α

From here, we can see that it is a rich gets richer type process, which interpreted in the context of Dirichlet Process Mixture Models, would mean that most documents generated would aim to be of the same topic. While this may or may not be appropriate depending on the data being modeled, it still gives us an intuitive understanding of the whole generative process and beyond that, it illustrates how it is possible for new customers/documents are free to initialize a new topic if they so require it.

Hierarchical Dirichlet Processes

Now, we will move on the generalization of Latent Dirichlet Allocation, the Hierarchical Dirichlet Process[15]. Like LDA, the HDP also generates topics and a distribution over them in the first step of the generative process. Then, each document draws its own distribution over topics from the distribution obtained in the previous stage. Each word in the document is generated in much the same way as it would have in LDA. Again, where they differ is in the number of topics in both cases. In LDA, the number of topics in finite while in the case of the HDP, the global pool of topics is of potentially

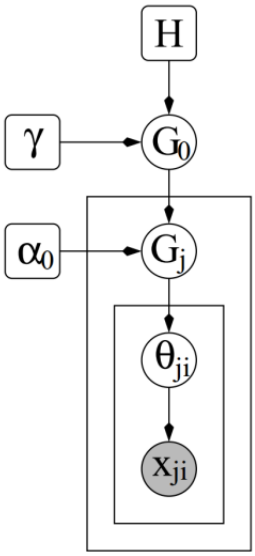


Figure 4.3: Plate Model of the Hierarchical Dirichlet Process

infinite size which allows each new document to sample new topics if existing ones do not model them well enough.The generative process for HDPs is as follows:

- A global pool of topics, G_0 is first generated from a Dirichlet Process with H as the base distribution and γ as the concentration parameter. $G_0 \sim DP(\gamma, H)$
- Each document, d_j now samples a document specific distribution over the topics, G_j by in turn sampling from a Dirichlet Process using G_0 as the base distribution and 0 as the concentration parameter. $G_j \sim DP(\alpha_0, G_0)$
- Each word, w_{ji}, then samples a topic, θ_{ji}, from G_i . $\theta_{ji} \sim G_j$
- A word is then generated according to the distribution of the topic that is drawn for that position. $w_{ji} \sim Multinomial(\theta_{ji})$

In this model, the fact that G_0 is discrete ensures that teach document will share the same set of topics. This solves the problem we wouldve had if G_j had drawn directly from H. The advantage of Hierarchical Dirichlet Processes over Dirichlet Process mixture models is that now, each document has a distribution over topics rather than having a single topic which would result in fewer topics being used. The fact that any topic found for a word

in a document should also have been drawn from G_0, the constraints on the posterior distribution over G_0 which is a Dirichlet process will ensure that not too many topics are drawn at all. The Hierarchical Dirichlet Process also has an intuitive sequential understanding analogous to the Chinese Restaurant Process called the Chinese Restaurant Franchise.

Chinese Restaurant Franchise

The Chinese Restaurant Franchise involves a group of restaurants instead of the single restaurant found in the Chinese Restaurant Process. Each restaurant has a set of customers. The catch is that each restaurant is part of the same franchise. In this metaphor, the documents play the role of the restaurant and words in the document play the role of the customers in contrast to the CRP where the documents were the customers and there was a single process. The enforcement of the franchise rule comes because each document samples topics from the same base distribution, G_0 and this ensures that topics are shared across documents. The topics in this metaphor rather than playing the role of the tables, now play the role of dishes served at tables and these dishes can be served at multiple tables at multiple restaurants.

We will now describe the Chinese Restaurant Franchise. Assume that there exists a Chinese Restaurant Franchise with a global menu common to all of its restaurants. Each restaurant has a local menu of the dishes that it serves.

- A new customer on entering restaurant, j will search for a dish of his liking on the restaurants current menu. This corresponds to looking at the dishes that ones whove come before him have taken.
- If he chooses a dish that an existing customer has already taken, he sits down at the table corresponding to that customer.
- If not, he consults the franchise menu for a dish. This will correspond to one of the dishes that have ever been served by this franchise. If he finds one he likes on the global menu, he then proceeds to create a table for himself and adds the dish to the menu of the restaurant he resides in.
- If even then, the customer doesnt like the dish, the franchise creates one for him and adds it to the global and the local menus for that restaurant.

This gives an intuitive sequential generative view for the Hierarchical Dirichlet Process and also makes it clear how topics are being shared across documents and when they are created. Now that we have covered all the

required preliminaries, we will now go on to describe how Entity Disambiguation can be done with topic models.

Chapter 5

Entity Disambiguation with Topic Models

5.1 Entity Disambiguation with Hierarchical Topic Models (Kataria et al)

Entity Disambiguation with Hierarchical Topic Models by Kataria et al was the first paper to seriously combine the topic models with knowledge bases for entity disambiguation, and at the same time use the hierarchical categorical information provided by wikipedia. They use the wikipedia category hierarchy for improved entity disambiguation by incorporating topic correlation between the topics assigned to the words in a document. We will spend a considerable time on this paper because this forms the base of a lot of what we will discuss later.

5.1.1 Leveraging topic co-occurence patterns

The basic idea is that certain topics appear together and disambiguating onc reference should help us to disambiguate other mention phrases in the current document. For instance, if Michael Jordan and LeBron James appears together in a document, then even in the absence of word-level correlation by basketball-related terms, we should be fairly confident while disambiguating Michael Jordan because of the presence of the popular basketball player LeBron James.

5.1.2 Why wikipedia?

Wikipedia has in-built meaningful correlation between topics (how much of the path is shared between the two topics). So if paths become topics for a document, then wikipedia provides a natural idea of correlation between topics. Also these paths would actually end in an entity and hence

these topics would actually correspond to entities too (though two different topics(paths) may correspond to the same entity).

The other advantages of Wikipedia is that it has both a huge bunch of entities (of the order of 3.4 million). Each wikipedia entity has a separate page and there is an incredibly huge network of links and annotations in Wikipedia. The huge amount of annotations serve as very valuable training data for the disambiguation process. And the best thing about Wikipedia : It's free!

5.1.3 Model

The generative process for the document is as follows :

- 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 $wi \sim Mult(\phi_{zili})$

5.1.4 Inference

We can use Gibbs Sampling to compute the topic path assignments z for document words in W. Gibbs sampling draws from the posterior probability distribution of P(z—w) by sampling topic paths for each words sequentially. The gibbs sampling equation is given by

$$P(z_i = \langle k_1, k_2, \ldots, k_l \rangle | \vec{z}_{-i}, \vec{w}, \{\vec{\alpha}_k\}, \vec{\beta}, \mathcal{A}) \tag{3}$$
$$\propto \frac{n_{t,-i}^{(k_l)} + \beta_t^{(k_l)}}{\sum_{t'=1}^{T}(n_{t',-i}^{(k_l)} + \beta_{t'}^{(k_l)})} \cdot \prod_{j=1}^{l-1} \frac{n_{k_j k_{j+1},-i}^{(m)} + \alpha_{k_j k_{j+1}}^{(m)}}{\sum_{k' \in c(k_j)}(n_{k_j k',-i}^{(m)} + \alpha_{k_j k'}^{(m)})}$$

5.1.5 Pruning and Blocking

The original wikipedia graph has a lot of problems. To begin with, the graph is not even a DAG and there are cyles in the graph. Apart from that, a lot of nodes in the category graph of Wikipedia are spurious and don't add any value to the wikipedia graphical structure and hence can be removed. For the purpose of pruning, Kataria defines the entropy of each node and then

removes the node with minimum entropy (Higher the entropy, the better the node is). Entropy for a node is defined as below :

$$E(k) = \sum_{m \in W} f(n_{k1}, n_{k2}...)$$

where f(a1, a2, ...) = $entropy(a1/\sum a_i, a2/\sum a_i, ...)$

The above notion of entropy tries to capture the correlation between children for a particular ndoe.

Since kataria is working at scale-web thefefore, we need to do some scaling. In the current case, Kataria tries entities in the entire tree before deciding the entity for a particular word in the document. This method is not efficient as a lot of combinations need to be tried. However, in general only a few entities are relevant to every document. And these are the entities whose surface form matches a key-word int he document. (The surface form of an entity are the anchor text appearing within links to that entity)

Chapter 6

Wikipedia Disambiguation Algorithm

6.1 Introduction

Existing work in entity disambiguation with topic models has sought to only disambiguate the mention phrases to existing entities.However, new entities always get added to the corpus and our knowledge bases might not always be upto date. Also the knowledge-bases don't usually bother with less-known entities. Also, when we expose our models to other test documents, then we might benefit from recognizing the new entities which these documents talk about (clustering these new entities). So we are looking at a generative process which also gives us the freedom of adding new entities during entity disambiguation.

To this end, we have proposed a non-parametric topic model which make use of the well-known dirichlet process in modelling the addition of new entities. We have started with a simple model (which has a lot of flaws), and built on top of it to finally create a robust model.

To achieve this, we have to change the category hierarchy of wikipedia slightly. The reason for this change will become apparent as we look at the modifications made to the model to accommodate new entities. The change is the following : For each category node X, we will add a new category node Y as the child of X and all the leaf nodes(entities) of this X will be moved to the new category node Y. So now we have three types of nodes in the tree. Category nodes, these pseudo-nodes which just take all the leaf nodes attached to the corresponding category node, and of course the leaf nodes. Let us call all these new category nodes special-category nodes.

If you just want to read about the final model which has been proposed, then move to section 4.

6.2 Using a CRP

We want to have the freedom to add new entities to existing categories. So dirichlet process comes to the rescue. Basically we want to change the generative process to give the special category the freedom to add new entities (Note that we are not looking for a model which allows the addition of new category nodes. And hence category nodes will be untouched). It's only the special-category nodes to which we will add new entities or leaf nodes. Let us look at the generative model for the same :

1. Generate a distribution β over infinite set of entities for all the special-category nodes

   $$\beta_n \mid \gamma \sim DirichletProcess(\gamma, H)$$

   for all special category nodes

2. Sample a word distribution for each of these infinite entities, for all the special-category nodes using a dirichlet distribution

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

3. For each document

   (a) Sample a distribution over each of the category nodes using a $dirichlet(\delta)$

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

   where x is the number of children of the category node

   (b) For each word in the document

      i. Start at the root node.

      ii. Sample a child using

      $$z_t \sim Mult(\pi_n)$$

      iii. Repeat (ii) till you reach a special category node

      iv. Draw the leaf node(entity) using the multinomial β_n where β_n was the distribution generated from the dirichlet process

      v. Draw the word using

      $$w \sim Mult(\phi_k)$$

      where ϕ_k is the word distribution for the entity chosen in the previous step

The purpose of adding a special-category node should be more clear now. If we had not introduced special-category nodes, then defining the generative model without defining a generative process for the category nodes becomes very difficult. So if we define a generative process over the category nodes for the purpose of leaves, then how would we have explained the child category nodes of the given category node? To avoid this problem, we have chosen to add this construct of special-category nodes.

6.2.1 Problems with this

There are quite a few issues :-

- New entities will be not be uniform with the previously-existing entities because new entities will have single parents while existing entities had multiple parents and this is not nice from the point of the view of a robust topic model.
- This is related to the previous point. Since entities cannot have multiple parent, this might lead to addition of the same entities multiple times in different special-categories (Suppose a new cricketer X appears in a list of ranji players, and then also appears in another document in which cricketers are talked about, then the player might get added in two different categories of "list of ranji players" and "cricketers of India" if the common path between the two is not long enough. I can see such problems happening in a variety of situations).
- We would like entities belonging to a particular topic to have similar word distributions. There are reasons for this. We worry that if this is not the case, then new entities will just get added to the category corresponding to the categories which they most often co-occur with. For instance, the wife of Sachin Tendulkar will just get added as a cricketer, because whenever sachin's wife appears, she appears in the context of cricketers (i.e. Sachin Tendulkar). So even thought the word distribution is giving us enough hints that Sachin's wife should go in a different category, we don't take the cues in the current system.
- How do we constrain these leafs to correspond to entities? If we are giving the freedom of adding new entities, then I might very well add leaves which correspond to say, the functional words . How do I ensure that only meaningful new entities are created?

Let's look to solve some of them using a more complicated model in the next section.

6.3 Using a HDP

Some of the above problems which come up for the CRP case can be solved by using a HDP model for entities. The idea is that when a special category wants to add a new entity, then it can add a link to one of the existing entity, or create a new entity altogether. We can have a two stage process to get the new entities. Basically the generative process for the entities is this : First we get a subset of entities using the dirichlet process which is shared globally, and then each special category gets a distribution out of this subset that we created in the first process. This is basically the application of HDP (Hierarchical Dirichlet Process) in our problem.

6.3.1 The Generative Process

Only the generative process for the topics change. So, the category hierarchy is fixed, and we generate the entities in the following manner . The parameters to the model are $\alpha, \gamma, \delta, \eta$

1. Generate a distribution β over infinite set of entities which will be shared across corpus

$$\beta_n \mid \gamma \sim DirichletProcess(\gamma, H)$$

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

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

3. For each special-category, sample a distribution over entities using a

$$\pi_n \mid \alpha, \beta \sim Dir(\alpha, \beta)$$

   where β was sampled globally and is a distribution over countable-entities

4. For each document

   (a) Sample a distribution over each of the category nodes using a $dirichlet(\delta)$

$$\pi_n \mid \delta \sim Dir(\delta, \delta, ..xtimes)$$

   where x is the number of children of the category node

   (b) Use the distribution over the entities for special-category nodes which was sampled globally

   (c) For each word in the document

      i. Start at the root node.

ii. Sample a child using

$$z_t \sim Mult(\pi_n)$$

iii. Repeat (ii) till you reach a leaf node(entity)

iv. Draw the word using

$$w \sim Mult(\phi_k)$$

where ϕ_k is the word distribution for the entity chosen in the previous step

Note(may be read optionally) :-
We decided to share the distribution across entities for special-category nodes globally. Would it have been a problem if we had sampled distribution across entities for special categories independently for each document? Yes. The most important problem that would have come up is that when we would have dealt with new documents, we would have lost the information about which all entities were associated with special category nodes in the original wikipedia hierarchy. This is because all the distribution over special-categories is independent for every document. So the correlation of the topics is not at all captured in the model because there is no globally shared distribution which captures this. So, now given a new document, the topic model will simply go to the same special category for every word, and will add all the necessary entities to this particular special category.

6.3.2 Problems with this model

There are several problems. We have solved the first two problems that came up in the previous case. However, the third and fourth problems described in the previous model remain unsolved in this model too.

A new problem with this model is as follows. The documents will no longer have an independent distribution over the children for a special node, but that distribution will be global. We have lost the independence of picking up an entity for a special-category in a document independent of other documents. This might be problematic and undesired. We don't really want the probability of picking up Sachin Tendulkar in the cricket category to be universally high for all documents. We would perhaps like this to be dependent on the document. (If we are talking about Indian cricketers, then we want this distribution to be high; low otherwise). From the generative side, it absolutely doesn't make sense that Obama has equal probability of being used in every document, when we reach the politics category. Real documents definitely have a separate distribution across entities. From the gibbs sampling side, it makes sense to increase the probability of seeing a topic again, if we have seen it once. This will not happen if the distribution across entitites for special-categories is global.

6.4 The final model : A three-level hierarchical model

How do we solve the problem raised above, wherein, we want the documents to have a separate distribution across entities for the special- category nodes? A natural solution can be to run a Dirichlet Process on this distribution over children for the special categories. This means that for a particular document, we will have to sample from the distribution across children for the special category, which was drawn globally. So, we have added an extra-level to the hierarchy of the dirichlet process.

The category hierarchy is fixed, and we generate the entities in the following manner . 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.

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

4. For each document

   (a) Sample a distribution over each of the category nodes using a $dirichlet(\delta)$

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

   where x is the number of children of the category node

   (b) Sample a distribution over each of the special category nodes using

   $$\pi_n \mid \zeta, \beta_n \sim DirichletProcess(\zeta, \beta_n)$$

   where β_n was sampled globally for the special category node

   (c) For each word in the document

      i. Start at the root node.

      ii. Sample a child using

      $$z_t \sim Mult(\pi_n)$$

iii. Repeat (ii) till you reach an entity/leaf

iv. 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

6.4.1 What does each of these hyper-parameters mean and what kind of values should they be given?

γ The propensity of creating new entities globally.

α There are two aspects to this parameter

- The propensity of creating new entities for special categories. Some categories have a naturally high tendency to add new topics and they can be given higher values(Actors, Politicians etc). However, categories with lower propensity for adding new topics can be given lower values (Prime Ministers of India, Nobel Prize Winners etc). This value can be heuristically set by looking at the number of entities added to a particular category across two wikipedia dumps (dumps for two different periods).
- This is also the variance of the distribution for the different special-categories. Since different special-categories are likely to correspond to different entities, therefore, we would like to set a generally high value for this hyper-parameter.

Hence we would like a generally high value for this hyper-parameter, but relative values can be decided by the rate at which entities are added to this special category.

ζ The propensity of adding new entities for the special category for each document. Also the variance of the distribution for the document-specific distribution over entities for a special-category. We would like to set a moderate value for this parameter, because the document-specific distribution aren't likely to be too different from the overall distributions for a particular category.

δ The dirichlet parameter for the distribution across the children for each category

η Simplest of all. The dirichlet parameter for the word distributions for the entities.

6.5 Inference

6.5.1 Chinese Restaurant Something

Well, the inference problem is now different from that of HDP because of one additional level. Let us look at the inference problem in details. We can first find an equivalent process for our WGP. The metaphor is as follows. There is an international restaurant franchise, a national restaurant franchise and then individual resaturants. Now when a customer(word) goes to a restaurant, it has two options. It either joins an existing table or creates a new table. When it creates a new table, the table goes as a customer to national level restaurant for this dish. Now at the national level restaurant, again there are two options. Either join an existing table and enjoy the dish which is being served at that table or go and create a new table and this new table goes and seeks out a new dish at the internation franchise. At the internation franchise, the same process repeats, but this time when a new table is creating, it just creates a new dish when-ever required.

6.5.2 Gibbs Sampling Algorithm

Now we need to think about the Gibbs sampling algorithm that will work for the above inference scheme. Here again, the gibbs sampling algorithm goes in the similar lines as the sampling algorithm in HDP. We shall have index variables t_{ji} (represents the index of the table for the ith word in the jth document) , k_{jt} represents the index of the national table for the restaurant based table and d_{ct} representing the index of the international table for the national table for each nation (i.e. supercategory)

Now, the inference equations are as follows : The conditional distribution

$$p(t_{ji} = t|t^{-ji}, k, d) \propto \begin{cases} n_t^{-ji} f_{k_{jt_{ji}}}^{-x_{ji}}(k_{jt}^{ji}, x_{ji}), \text{ if t has been used previously} \\ \gamma p(x_{ji}|t^{-ji}), \text{ if t= } t^{new} \end{cases} \tag{6.1}$$

If the sampled value of t_{ji} is t^{new} , we obtain a sample of k_{jt}^{new} by sampling from :

$$p(k_{jt}^{new} = k|t, k^{-jt^{new}}, d) \propto \begin{cases} m_{\cdot k} f_k^{-x_{ji}}(x_{ji}), \text{ if k previously used} \\ \alpha f_{k^{new}}(x_{ji}), \text{ if k = } k^{new} \end{cases} \tag{6.2}$$

If the sampled value of k_{jt} is k^{new}, we obtain a sample of d_{ct}^{new} by sampling from :

$$p(d_{ct}^{new} = d|t, k, d^{-ct^{new}}) \propto \begin{cases} l_{\cdot d} f_d^{-x_{ij}}(x_{ji}), \text{ if d previously used} \\ \zeta f_{d_{ct^{new}}}(x_{ji}), \text{ if d = } d^{new} \end{cases} \tag{6.3}$$

Now when we have have to sample a d^{new} from a continuous distribution, then we will never get any repitions. So it's just safe to sample directly from the distribution. However, we can assign any id to this new sample as the id itself doesn't matter.

Now, once we have sampled this for a new word, we also need to sample the k_{jt} for different tables for documents, and d_{ct} for the different tables in the supercategory. Notice that this works out using the above equation only. However, while considering the second part of the RHS, we will need to consider all the words which will get affected by this sampling.

Note : We can also perhaps decouple the inference across documents (I was trying to work out the expression but I don't know what the posterior of DP process is if we have samples of the DP process. Note that the samples are infinite sized and hence they are described in terms of $\sum \beta_k \delta_{\phi_k} + \beta_u G_u$ where G_u is a sample from $GP(\alpha, \beta)$ where β itself is a sample from $DP(\gamma, H)$ (Ref : Section 5.2 in HDP paper)). However, the major issue is that I don't think that there is a motivation/reason behind trying to decouple inference across documents in any case. (In the case of HDP, the purpose behind decoupling was to use the inference algorithms in more complicated models at the expense of poorer quality of inference. Even when we distribute the inference task, the decoupled inference is perhaps not likely to be better off than the inference using the delayed updates).

6.6 What about the left-over problems?

6.6.1 Problem of prior

The issue was that how do we have some sort of prior over the word-distributions depending on the categories they are linked to? Because if we don't do something about it, then if a new entity doesn't co-occur with entities of its own category, then its categeory might be inferred incorrectly. So, there might be two type of cases here

- Entities which doesn't appear with other entities of its categories : For instance, J.K. Rowling might not occur regularly with other authors. There might be a few lists which list down the authors, but if they are not present, then it might be difficult to infer that J.K. Rowling is an author. This, despite the fact that the word distribution for J.K. Rowling might carry enough hints about J.K. Rowling being an author.

- Entities which are strongly correlated with another entity of a different category : For instance, Sachin's wife always appears in the context of Sachin.

We don't know what to do to solve this problem but we are not really sure of the number of cases in which it will be required. Maybe we don't really need

to worry about such cases as such cases might not be ubiquitous. Also, it's not such a big problem if there are some mistakes in the identification of the category in a few cases as long as entities get discovered with good precision. However, it will be nice if something can be done about this problem.

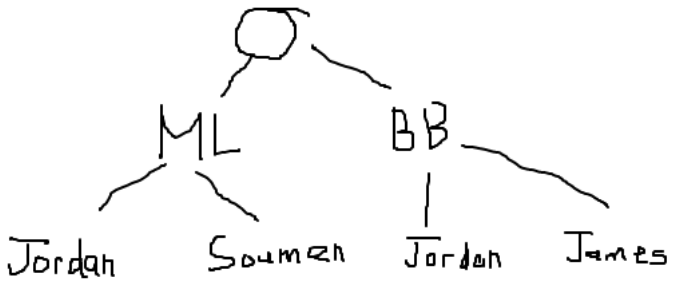
6.6.2 Problem of spurious topics

I had raised this problem while talking about the CRP. How do we stop the model from creating spurious topics. Topics for functional words or topics created from words which are correlated(appearing in similar context ; Example - words like corpus, documents, collection etc might form a topic which we don't want), or other spurious topics which do not correspond to entities in general. Do we need to constrain our topics in some way? We might not need to constrain it, because if the model wants to create topics for, say functional words, then it will have to create these topics for most of the categories. (Because we are also enforcing topic level correlation). Now doing this might be difficult for a topic model and hence the topic model might not choose to create these spurious topics. Basically the seeding of the topics with existing entities might help it to go in the correct direction.

6.7 Experiments

The above model was implemented on Java with the CRF-based gibbs-sampling inference algorithm. The experiment setup was

1. **Wikipedia Graph Structure** : For the initial testing, we have used a very simple DAG-category structure consisting of the root which has two children, Machine Learning and Basket-ball. Machine Learning in-turn is connected to the entity Soumen Chakrabarti and Michael Jordan and the basket ball category node is connected to Michael Jordan and LeBron James, two of the most popular basketball players of the current age. A visual has been attached



2. **Data** : We had a very limited data-set consisting of just a single training line for each of the four entities (the line is the first line of the entitiy page for the entity), a few manually written lines to see if the essential features of the model are being demoed by the test data and a few lines for Penelope Cruz, a celebrity. The idea was to see if penelope cruz would be clustered as a different entity or not. We applied stop-word filtering on the data and ran the inference algorithms.

3. **Stopping criterion for Gibbs Sampling** : Currently, we are running a fixed number of iterations (1000) and then stopping after that whenever we are within 3% of the log likelihood of the data of the minimum which was ever achieved during all the iterations.

The results were very good. The algorithm was able to achieve all the things that were promised

1. The algorithm was able to disambiguate well based on just the **word-correlations** (in sentences where there were no other related entities) : In the sentence "Michael Jordan has worked on Hierarchical Dirichlet Process which is an important value addition to the field of machine learning an artifical intelligence" , the algorithm could disambiguate Michael Jordan to the ML Researcher

2. Based on **topic-correlations** (in sentence where no related words were present, but related entities were present) : In the sentence "Soumen Chakrabarti and Michael Jordan", the michael jordan was correctly disambiguated to the ML Researcher again even though the only clue in the sentence is the presence of soumen chakrabarti

3. Able to discover new entities (successfully found out the presence of Penelope Cruz, Ganesh Ramakrishnan, and Khushi Ram in the data and created a new topic for it). The other interesting thing is that it also puts these entities in the right place in the wikipedia category graph.

    - Discovering Ganesh Ramakrishnan : On adding the following sentences to the corpus, "Ganesh Ramakrishnan, and Indian researcher does Machine Learning and mining like Soumen Chakrabarti", and "Ganesh Ramakrishnan is a professor at IIT Bombay in the field of Machine Learning and Information Retrieval. His work in the field of rule based methods have received appreciation around", the WDA creates new entity for Ganesh Ramakrishnan and attaches it to the Machine Learning category node
    - Discovering Khushi Ram : For those unaware, Khushi Ram was an Indian basketball player and is regarded as one of the all time

greats of basketball in Asia. On adding the following sentences to the corpus, "Khushi Ram, an Indian basketball player from the small town of Khushra has found national acclaim. A new player Khushi Ram is being praised over the world and people say that he is the Lebron Raymond James of America. Khushi hails from the small village of Khushra", and "Khushi Ram has been making news again with his exquisite skills", WDA creates a new entity for Khushi Ram and puts it in the node with MJ and LeBron James

The results are very encouraging and definitely call for further experimentation with larger scales of data. The experimentation also validated the apprehensions we had, where in it created a topic which didn't correspond to any entity and also attached one of the topics corresponding to penelop cruz to the machine learning node. However, these things happened mostly because the data was too less to do anything meaningful. It would be interesting to see how it performs with larger scale of data.

6.8 Power of the Model

So the above model is powerful and achieves the following :

- It can use the knowledge-graph of wikipedia effectively to model documents and topics
- It can capture topic-level correlation in documents. And of course, it can capture word-level correlation of topics.
- It can add new entities to the model as and when required. Hence we are not stuck with only the entities that wikipedia has to offer. The new entities can have multiple parent category nodes, and hence these new entities are just like any other entity of wikipedia.
- It does all the above in a very principled way.

The model can be used in various ways :

- The model can be used to discover new entities which are not present in Wikipedia in a new document
- The model can be used to augment the wikipedia category strucuture, by adding new entities at the correct place
- The model can be used to discover irrelevant category nodes in the wikipedia graph structure (categories like "list of people born in 1952)

6.9 Challenges and Future Work

6.9.1 Learning the hyper-parameters

We will have to devise techniques to reach good hyper-parameter values. This can be done using cross-validation or putting another prior on top of the hyper-parameters (which will increase the complexity of inference though)

6.9.2 Gibbs Sampling stopping criterion

Currently, we are running a fixed number of iterations and then stopping after that whenever we are within 3% of the log likelihood of the data of the minimum which was ever achieved during all the iterations.

6.9.3 Running it with Wikipedia

There are a number of challenges with running it on a wikipedia-like scale. The first set of problems come up because of the graph structure of wikipedia.The wikipedia graph contains cycles, contains a lot of meaningless nodes and so on. Which means that we will have to meaningfully prune the wikipedia graph to keep only that which is essential. Kataria et al have discussed some techniques for the same which we can use for our work.

6.9.4 Good blocking techniques

We will need to find good blocking techniques to identify the important entities for a given document. Kataria talks about selecting those entities whose surface form matches a keyword in the document. We can use the same or can also look to improve this heuristic.

6.9.5 Scaling up the inference

We cannot really just work with the feature vectors when we are working in a topic-modeling setup(if we could then we could do some sort-of distribution of the feature vectors formed by different words). There are two directions which we can take :

- Distribute only the task of inference with all the documents being in a common place. Take a feature vector out, send it to one of the nodes (the node corresponding to this particular mention phrase), make it do the inference task, and bring it back. This will involve a lot of message passing.
- However, if we are distributing the documents itself, then we will have to do some sort of delayed updates. However, we can do some sort

of smart distribution which minimizes the effect of delayed updates. Basically, we try to group documents which have an effect on one another in a single node (the counts are updated inside a node and any change within a node is immediately reflected and can be accounted for by other documents). I had worked on precisely this sort-of a problem in my internship(for scaling up HDP) and have some ideas regarding this issue (Basically cluster documents which are likely to have similar entities together using some similarity measure for documents(Jaccard Similarity is a candidate similarity measure))

6.9.6 Using other features

There are a lot of features which we have missed out on. A wide variety of useful information is out there, waiting to be used.

- How do we account for the entity page itself on wikipedia? It makes less sense to actually ignore its special properties in disambiguation.
- One of the most important feature for entity disambiguation is the anchor text matching with the wikipedia title. However, currently this is being given no importance. I mean, if I see an entity X Y and there is just one X Y in wikipedia, there is a very high likelihood that it should be this very X Y only. Why bother with only word correlations and topic correlations only when there are so many other clues to be picked up?

Incorporating other such features in topic modelling algorithms is usually a painful process as there is no principled way to do so.

Chapter 7

System Design

In this chapter, we will discuss the design of our entity linking and disambiguation system along with the preprocessing steps required to get it working. First, we will motivate our choice of using Wikipedia as a training data set and as a source for the required hierarchy and also describe its relationship to the Knowledge Base provided by the Text Analysis Conference as part of its Entity Recognition and Linking challenge which we will use as the primary test set for our model. We will then describe how we preprocessed the Wikipedia Category Graph, first to make it a Directed Acyclic Graph and then to speed up inference keeping the TAC KB in mind. Next, we will propose some heuristics for improving the performance of the system both in terms of speed and accuracy. We will then explain the reasons behind us choosing to use the YAGO category graph for extracting a smaller training set for our experiments.

7.1 Why Wikipedia?

Wikipedia is a free-access online encyclopedia that anyone can access and edit. It is comprehensive about its coverage of real world entities. Also, due to its crowdsourced nature, it is constantly being updated about facts relating to real world entities. Also, the organization of Wikipedia into pages with each page describing a particular entity make it a rich source of reliable information about that particular entity. Thus, entity disambiguation systems benefit in two ways by using Wikipedia pages for disambiguation:

1. Wikipedia contains a large number of entities making it more likely that the referent of a candidate mention phrase lies in Wikipedia
2. Wikipedia has up to date information regarding the entities in its knowledge base. This can only aid disambiguation systems as they are more aware of the current context of that entity

In addition to the use of Wikipedia entities as targets for disambiguation systems, Wikipedia also has an extremely rich category structure which also provides useful information about disambiguating mentions. A rich category structure like the one provided by Wikipedia, have been exploited to augment less frequently occurring entities with new information by exploiting other entities sharing the same category structure as seen in [4]. In addition to this, one would also expect that the category graph is indicative of the relationships between entities of various entities. That is, entities that share similar categories are expected to occur frequently together than ones that are from different categories. This can be used to disambiguate mentions if other mentions have already been disambiguated as the categories of the disambiguation mentions can give cues to the referent of the first one.

We will now motivate our choice of Wikipedia from the point of view of our test set, the test data from the Entity Discovery and Linking Challenge(ERL) from the Knowledge Base Population(KBP) track at the Text Analysis Conference(TAC).

The ERL challenge is the first to introduce evaluation metrics which are used to measure performance on the entities not in the Knowledge Base. All earlier editions of the ERL and ERD(Entity Recognition and Disambiguation) challenges tested only for mentions which were present in the KB. From the 2014 edition of ERL, there will also be an evaluation of the ability of disambiguation systems to cluster NA entities. This is precisely the kind of test set that we would like to run our model on and thus makes it an ideal choice for evaluating our model.

Now, how are the Knowledge Base used in TAC and Wikipedia related? The knowledge base used in TAC is created from an October 2008 dump of english Wikipedia. Therefore, some of the reasons behind our choice of Wikipedia are:

1. Wikipedia articles give good coverage of the entities in TAC while also simultaneously providing valuable information about the entities themselves

2. The KB used as part of TAC does not come with its own category structure which is central component of our algorithm

3. Wikipedia is also a large source of annotated documents that can be used to train our model as the ERL challenge itself does not provide adequate documents for training our system

Despite all the positives of using Wikipedia as a source for training data and the category structure, there are still disadvantages to using it, one of which is the problem of having a defective category structure, which we'll discuss in the next section.

7.2 Parsing and Pruning Wikipedia

In this section, we will discuss how we parsed Wikipedia documents and pruned the Wikipedia Category Graph to make it amenable to our type of disambiguation model.

Parsing Wikipedia Documents

Parsing Wikipedia documents was something we wasted a lot of time on while we were trying to parse the raw wikitext pages ourselves. Thankfully, we found out eventually that JWPL also had an interface through which parsed the documents as a parse tree which could be traversed by writing an appropriate Visitor class.

Pruning the Wikipedia Graph

We used the software package, JWPL, to process the raw Wikipedia dumps which are obtainable from the Wikipedia website[1]. This will provide us with a convenient method to access and use the Wikipedia pages and the Category information. Unfortunately, this does not solve the problems with using the Wikipedia category structure in its raw form. The Wikipedia category structure is spurious with cycles and self-loops whereas our algorithm requires a DAG with a single node at the root to work. We will now describe how we pruned the edges in the Wikipedia Graph to make it a DAG.

Firstly, we need to add a root node to our graph. We need to add this node while making sure that every node in the graph is reachable from the root node. On the other hand, we would like to add the minimum number of edges to make sure that the whole graph is reachable from the root node. A naive way to do this would be to add a root node and connect it to all the nodes with now incoming edges. This is not a feasible solution as a standalone cycle would still remain unreachable from the root node. The following algorithm is the one we used:

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
3. Connect the root node n_r to one element in each of c_i for $i \in 1 \dots k$

It is quite easy to see that the number of edges added in the above algorithm is optimal if a root node has to be introduced.

Now, we need to ensure that the category structure thus formed is DAG. Again, we need to think about the kind of edges that we would like to retain

[1] https://dumps.wikimedia.org/

in the graph. While several heuristics may be proposed, the one we went with was to retain the edges that keep the nodes closest to the root. We would not want our nodes to move farther because more information is lost when an edge close to the root is lost rather than when a far away edge is lost. If the edge is far away, few documents will be affected severely as not many documents will have a topic that uses a node that is far away from the root. On the other hand, several documents might lost out if an edge between nodes at the top of the hierarchy is lost. In addition, the documents that do make use of far away edges already have significant topic information along the path that leads to the edge. So, even they do not lose out significantly. We now propose the following algorithm to make our graph a DAG.

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

This algorithm guarantees that *all* shortest paths from the root to all the nodes in the graphs are retained. So, not only does any node not move farther away from the root, all the paths from the root to the node which were the shortest are retained.

Now that we have pruned the Wikipedia graph to remove cycles, it might seem that we will be able to run our algorithm on the training set which we have chosen to be Wikipedia. While we can theoretically run the algorithm using the graph that we have so far, it isn't practical to run the algorithm on the Wikipedia graph we have now. The graph at this stage has about 900000 nodes. Running the algorithm on this graph with more than 800000 entities from the TAC Knowledge Base and more than a million training documents is out of our reach.

One solution to this problem is to prune the Wikipedia category graph even further. And from the experiments in [16], we see that even 12 nodes ensure good performance with close to 60000 entities. We, therefore, can afford to prune the graph heavily.

We used an algorithm developed by Ramakrishna Bairi, a PhD student working with Prof. Ganesh, whose work is on selecting categories for a group of entities in the context of disambiguation pages. We ran the algorithm on the $800,000$ entities in the TAC KB to get a set of 1000 nodes that properly

characterize the entities in the TAC KB. We will call these 1000 nodes, Salient Nodes.

But once we have these 1000 nodes, we still need to extract a complete graph from these nodes. We use several heuristics to obtain a smaller graph:

1. We first remove all nodes which cannot reach any of the Salient Nodes

2. We then remove all the nodes with only a single outgoing edge in the reduced graph
   These types of nodes are useless for the disambiguation process anyway. We will connect the parents of this node to the child of the node. If the child is also to be pruned, we take the single outgoing edge recursively till we hit a node which won't be removed. This process is guaranteed to terminate since our graph is a DAG.

3. We then remove nodes for which there exists a child which can reach the same number of Salient nodes as the parent.
   We take this to mean that the child which reaches the same number of Salient nodes as the parent already ties together all the salient categories that the parent covers. Therefore, we are safe to remove the parent and redirect the parents of the parent to the child. If this is not possible because the child is being removed, we proceed in a recursive fashion as before.

With the heuristics mentioned above, we managed to prune the graph from about $900,000$ nodes to just $3,400$ nodes. This is a tremendous drop and the running time of the naive algorithm is significantly sped up because of this pruning.

7.3 Heuristics for Speeding Up Inference

In this section, we will discuss some of the heuristics we've implemented to help speed up inference in our model. We have two main heuristics that we have implemented to improve the performance of our system:

1. Caching probabilities at the interior nodes of the graph
   This heuristic does not change the behavior of the model. It caches probabilities at the interior nodes of the graph and the nature of the model is such that these probabilities needn't be updated that often. This can lead to significant speed ups in cases where the graph is close to being a balanced tree.

2. Blocking heuristics employed to restrict the set of entities that can be sampled for a document.
   This heuristic unlike the previous one does change the behavior of the

model. This works by not allowing the document to sample topics which are extremely unlikely to be related to the document. A simple method doing this is by considering all the mention phrases and their candidate referents in a document and restricting the document to sample only these entities. A spotter outside the model is used to subset these entities for each document.

Caching Probabilities on the Interior Nodes

This optimization exploits the fact that once a path has been sampled for a particular path, only nodes along that path have their outgoing probabilities changed.

Ideally, we would like to have the following quantity precomputed at each node of the graph, $P(w|n)$. This quantity if available at every node can be used to compute $P(n|w)$. This is the probability of the word, w being generated given that we start from node n. This will make the sampling of paths easier by a logarithmic fraction. This graph, G, being a DAG allows for a recursive computation of probability, $P(w|n)$. Let n_c $c \in 1 \ldots k$ denote the children of the node n. The probability $P(w|n)$ can be computed recursively as:

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

Thus, if we have $P(w|n)$ for the lower nodes of the graph, G, it can easily be computed for the higher nodes of the G. Computing the value, $P(w|n)$ can be computed easily for the special category nodes. Let n be a special category node. Assume that β denotes the global distribution over entities, β_n be the global distribution over entities at the special category node n and β_{nd} by the document specific distribution over entities at the special category node n. Also, let ϕ_k denote the probability distribution over words for a particular topic k. Let k_d denote the number of distinct topics(tables) sampled for document d at node n with t_{id} denoting the topic of the i^{th} topic at node d and ϕ_t denoting the distribution of topic t. Thus, the total probability at node $P(w|n)$ can be computed as follows:

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

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

The integral in the second term in the second equation is common to all the special category nodes(note that its coefficients differ for the different

special category nodes). In a special category node other than the one currently sampled, none of the coefficients of the ϕs or the integrals will change. What can change however is the value of the integral in the second term of the second equation. But as we see that this is a global value it can be computed once for all special category nodes. The other three terms vary only because of the change in the topic distribution for the entity chosen during the last sampling step. We see that at the supercategory node, the probability is defined fully by a combination of the entities sampled at this node and the global distribution over entities. In general, the probability at a node $P(w|n)$ will be written as a combination of the topics that were picked by super categories which are descendants of node n and with a certain probability sampling the global distribution over entities. These coefficients will not change unless the node sampled in the last round is a descendant of n. Therefore, these coefficients will only have to be changed for ancestors of the node n' where n' was the super category node picked in the last sampling step. In a proper balanced tree, this leads to significant savings by a factor of $\ln(n)$

Figure 7.1 is an illustration of the values that will be maintained at each node. As we can see, they maintain the probability of reaching the global distribution over entities(the coefficient in the second equation) and the distribution over entities that its children can reach. If in case, the path $Root \rightarrow C2 \rightarrow C5$ is chosen in the previous step, the distribution over entities for the node $C1$ will not change. What will change however is the integral in the second equation over the global distribution over entities as its posterior might be affected by $C5$ sampling from it and also the distribution over words for the topic sampled at $C5$. The coefficients of these two terms however do not change for $C1$. Thus, the values maintained at $C1$ are all that are required to compute the value of $P(w|C1)$. This makes the sampling step for the next iteration really easy and reduces the cost of an iteration by bringing it down from a linear function of $|V|$, the number of nodes in the graph G, to a logarithmic function of $|V|$. So, in general, when the number of ancestors of most supercategory nodes is small, we can expect significant savings through the use of this optimization.

Subsetting Topics for Documents

Unlike the optimization in the previous section, subsetting entities for documents change the behavior of the model. This types of heuristics work by quickly subsetting the large list of entities(TAC has about $800,000$) to a much smaller set of relevant entities for the document(Maybe about 300 entities or so depending upon the size of the document). The previous optimization that we saw requires each node in the graph to store potentially number of entities parameters. This computation might be prohibitive if the number of entities is really large. Thus, subsetting the entities can lead

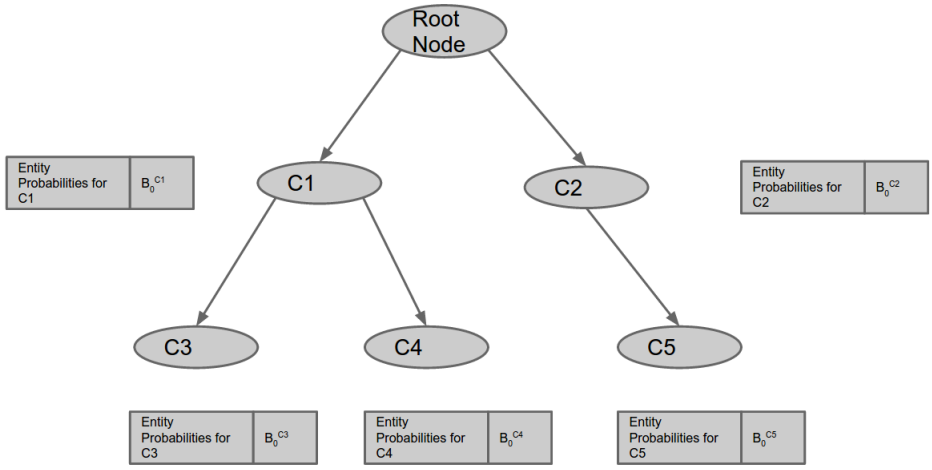


Figure 7.1: Caching Probabilities along the nodes C_1, C_2, C_3, C_4 and C_5

to significant savings depending on how aggressively the subsetting is done. And furthermore, the previous optimization and this one both improve on different aspects of the problem. The previous optimization alleviates the problem by reducing the computation that arises because of the graph structure while this one reduces computation that arises out of the number of entities in the Knowledge Base. One simple method to perform subsetting is to run an NER Tagger on the document to get a list of candidate mentions to annotate and collecting the list of all the candidate entities for all the mentions in the page. This list of candidate entities can be used as a subset of valid entities for this document.

Unfortunately, even these optimizations are still not sufficient to allow us to train multiple times on a dataset as large as Wikipedia. Also, the categories in the Wikipedia graph are quite noisy. It is for this reason that we decided to subset a smaller part of Wikipedia using the Ontology from YAGO for as the hierarchy graph instead.

7.4 Reduced Training Set with YAGO Hierarchy

To make experimentation easy, we decided to subset a smaller portion of the documents to quickly run the model and evaluate its performance. For this, we wanted to have a good category structure and also pick a set of entities in coherent topics. Towards this end, we turned to YAGO[17] for an ontology. YAGO's ontology/hierarchy is made from both Wikipedia's category graph and Wordnet. While, the Wikipedia category graph is very rich, it is also inconsistent. There is no clear definition of the relationship

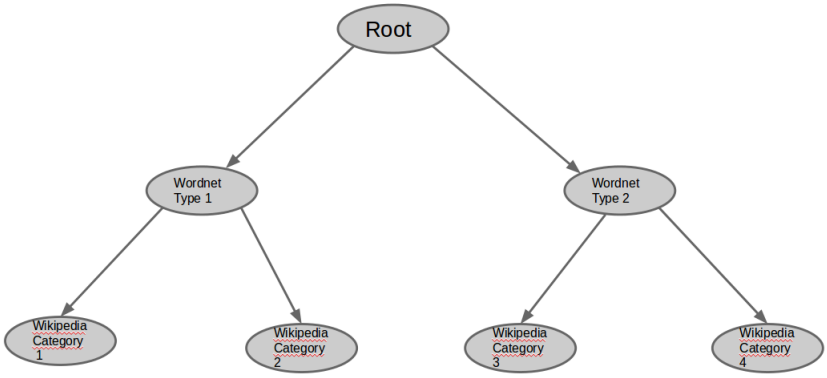


Figure 7.2: Illustration of YAGO Ontology

between categories. This leads to spurious category graphs with cycles and also a large number of noisy categories(20th Century Births is one such example). Wordnet on the other hand is much cleaner than Wikipedia with strict rules for defining relationships between words. See Figure 7.2 for an illustration of how the YAGO ontology can be constructued. For concreteness, the root node may represent the wordnet entry "sportsperson" with the two children being "cricketer" and "footballer" respectively. The Wikipedia nodes below that may be the category "Lists of Cricketers" for instance. Thus, YAGO combines the richness of the Wikipedia Category Graph with the cleanness of Wordnet's hierarchy.

For our experiments, we decided to subset the part of the YAGO Ontology which concerned scientists. We, therefore, traversed down the YAGO Ontlology from the Wordnet node with scientists and extracted a meaningful category graph due to Wordnet. Scientist had children for physicists, biologists, etc. This alse resulted in a reduced entity set of about 6000 entities and a reduction in the number of documents to about 5000. We use this set of data to quickly perform experiments and tune the hyperparameters.

7.5 Scaling Up

Once we have finished testing our model, we will be able to run on the whole of Wikipedia for training. The optimizations given above will reduce the time for a single iteration to about an hour. This brings the training time of the algorithm within the limits that we can tolerate. If the model doesn't work or the algorithm doesn't scale, we have discussed several methods of scaling up the inference in 6. We also introduce some structure learning methods to learn these hierarchies where we have explicit control over complexity of hierarchies.

Chapter 8

Experimentation

We plan to run the training on the entire 800,000 TAC entities and 4 million wikipedia documents so that we can evaluate on the TAC evaluation set. However, as we had motivated intially, there are tasks like learning the hyper-parameters, understanding the model and introducing changes which cannot be done with a very large dataset because otherwise each iteration will take up too much time and the process will become very tedious and time-consuming.

8.1 Theoretical estimate of the running time

We can do a theoretical estimate of the running time and it turns out that the estimate is pretty close to the actual time taken by the real implementation.

Let's first calculate the time taken by each word for the inference process. So, each word calculates the probability of picking up that word on one of the paths (which was rendered invalid by the previous word). Apart from that, it has all the probabilities pre-computed. At each node, to find the probability of the word at a node, it just uses the entity coefficients and multiplies it with a quantity which depends on the count of the times a word has been generated by the entity. Hence, the complexity of inferring on every word will be close to the

$$\mathcal{O}(num_entities * depthofthetree)$$

This entire process is repeated for all the words in the document. For each document, there is a significant set-up cost when all the probability coefficients etc are setup.. However, it's a one-time cost and can be ignored for overall cost estimate. So total time taken for a document

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

Multiplying it by number of documents and number of iterations brings the order to

$\mathcal{O}(num\_document * num\_iterations * num\_words * num\_entities * depth of the tree)$

Now, when we ran experiments on the first 500 documents, we had the following nature of the data :

- number of words = 350 per documents
- number of entities = 1500 (with YAGO entity set)
- depth of tree = 5 (YAGO graph)

So, the expected time taken is 0.05 seconds per document (without accounting for the constants). In reality, the average time taken with the entire set of YAGO entities is close to 0.6 seconds per document and hence the constants are about 10.

However, when the number of entities are very few per document (for instance, when the entity subsetting is active), then the initial setup time and other constant times start to dominate.

8.2 Training and testing framework

8.2.1 Selecting the document set for the given entity set

Now that we have a reduced entity set, inferring on the entire wikipedia wouldn't make sense. We would like to train on the document, which actually talk about the set of entities that concerns us and then learn the distributions from this set.
We can set a filter on the number of annotations of the relevant entity set as the parameter for selecting a particular document. However, that leads to poor results as it ends up selecting large documents which mentions relevant entities a couple of times, but ends up mentioning irrelevant entities a lot. Hence, a better metric to select (or not select) a document is the relative occurrtences of the relevant entities. We find the number of times relevant entities occur in the document. And the total number of mention phrases in the document (we just find the number of capitalized words to approximate this number) and take a ratio of this and take all documents whose ratio is above a particular number(0.25 is the number which we have chosen for now).

8.2.2 Frame work for testing

Now, to test the documents, we are picking up every 10th document and hiding the annotations for that document. After that, we run the usual disambiguation algorithm and in this process, the inference algorithm is

also run on the test documents (The inference process is oblivious of the annotations for the test documents). Now once this is done, at the end of inference process, we can just see if the entity which were assigned to the words were the same as the original annotation which we had seen for the word (for those words for which some annotation was present originally).

Also note that we are only testing on entities on which we had trained in the training dataset, and we just ignore the rest of the annotations.

8.3 Current Experiment

8.3.1 Experimental Setup

Currently, we are running the experiments on a smaller entity-set (as explained before). The YAGO graph which we are working with currently has the following characteristics :-

- Number of entities : 6000 entities
- Number of nodes in the graph : 83
- Number of edges in the graph : 84
- Number of documents : 4953

It is important to note that when we subset documents in the way that we described, we got only 3590 entities which had at least one annotation in the documents. Rest of the entities didn't even appear. Even for the entities which appeared, most of them had very few annotations and only a small fraction of them were well annotated and had enough training data.

8.3.2 With 5k documents

When the number of subsetted documents is 5k, then 3.5 k out of the 6k entities are active.

Without entity subsetting

One iteration over 5k documents without entity subsetting take 50 minutes. The accuracy of inference is not so good because of sparsity of training data and also because we couldn't run too many iterations because of the long training time. 725 out of 4139 annotations were correctly marked.

With entity subsetting

With entity subsetting, this process happens in a flash. The entire disambiguation runs in about 6 minutes (10-15 times faster) for a single iteration. Again, the accuracy is not so good here. Only 439 out of 4139 entities were

marked appropriately. The ratio of speed up is not so good perhaps because a lot of entities are being created.

8.3.3 With 500 documents

When the number of subsetted documents is 5k, then 1.5 k out of the 6k entities are active.

Without entity subsetting

The time taken per iteration is 320 seconds. For a single iteration, it reaches an accuracy of about 200 correct annotations out of 786 annotations.

With entity subsetting

This process is extremely quick. It takes about 10 seconds to perform a single iteration. The average number of entities is 30 and hence we expect a performance improvement of 50 times and we are actually getting a performance improvement of 35 times which is good enough (The constants of the entity subsetting get added because of which we don't get 50x improvement). The results however are abominable and only 118 out of 786 were annotated correctly.

8.3.4 Effect of increasing the number of entities

Without entity subsetting, 1.5 k entities and 350 words/doc on average takes 0.6 seconds on average. Similarly 3.5k entities and 140 words on average per document again takes 0.6 seconds on average. So it means that the running time increases linearly with the number of entities.

8.3.5 Effect of number of iterations

The number of iterations obviously affect the number of correct annotations strongly. So with a single iteration, the number of correct annotations, with 500 documents and 1.5k entities was 174 entities. When we did 2 iterations, then the number of correct annotations increased to 200 entities out of the 786 annotations.

8.3.6 Sensitivity to hyper parameter setting

The model is extremely sensitive to hyper parameter setting which leads us to believe that it is extremely important to optimize to a good hyper parameter setting. As an example, when we changed the changed the eta parameter from 0.05 to 0.2, then the number of correct annotations fell from 200 to 69 with 500 documents and a single iteration.

8.4 Observations

8.4.1 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 entity in the subset of 5000 documents was just 12 (which means that about 3 or 4 annotations) and about 35% of the entities had just a single annotation out of the 3.5 k entities that were marked in the data set. And this is the case after annotation completion.
Another major problem with sparse data is that it becomes very easy to just direct the existing entities in wrong directions because they don't

8.4.2 Poor performance of blocking techniques

The code given to us by Ashish is performing terribly. On manual check on a few documents, they seem to give only 25% to 40% of the candidate entities for every document which is a very poor number. In any case, we need a customized code for us which can also provide us with candidate entities out of the new entities that were created.
We are planning to write our own code for this purpose which will definitely work well and will be customized to this database and setting.

8.5 Setup for future experiments

The entire setup for full-blown experiments with the TAC dataset is setup already. The wikipedia graph has been pruned based on the TAC entity set

- Number of entities : 800,000 entities
- Number of nodes in the graph : 3400
- Number of edges in the graph : 10000

We will begin experiments with the larger datasets soon now since every thing in the pipeline is setup. The expected time per iteration will be.

- 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)

Hence time taken per iteration will be 10^{11} seconds which is close to 10000 seconds, accounting for the 10x factor. Which is 2.5 hours. Now to cut it down, we can take the first paragraph of the document. If we run 20 iterations or so, then the entire training process will take 2 days to complete.

Chapter 9

Structure Learning Topic Models

In the following chapter, we cover some nonparametric models which, beyond just exploiting structured objects like graphs to model correlations, also learn the structure of these objects.

9.1 Why Learn Structures

The first question that needs answering is why learning structures is important. Why not contend ourselves with the DAG structure used here as part of the proposed hierarchical disambiguation model. There are several possible reasons for this.

- Unavailability of a structure - We may not always be able to find a structured object. We may have found one here but that doesn't mean this will always be the case.
- Structure is insufficient - The structure provided may not be suitable to the task at hand.
- Structure needs to evolve - Often data sets grow over time, and this necessitates changing the structure to reflect the information that new data brings in.

As we will explain later, some of these reasons will become relevant to our task of entity disambiguation.

We will now move on to describing two models which have been used to discover topic structures from corpora. While the topic hierarchies are different from those being considered here, they can still provide us with ideas to generalize the disambiguation model proposed previously.

9.2 Notation

In this chapter, we talk about a DP without explicitly defining its support. While this was easy to do in the models discussed previously where the support DP were over the set of possible topics, this is not so easy in the models discussed here. Here, we will have DPs whose support itself is the set of possible outcomes of a DP whose support is the set of possible topics. This can be looked upon as a tree where the output of one DP is a distribution over distributions over topics. Since, both distributions are over countable support, this gives the impression of an immensely large tree. This structure can be repeated as many times as required to get really large structures. We will avoid referring to such DPs as distributions over distributions . . . topics or in the case of nHDPs infinite distributions and rather look at them from the point of view of a graph where the nodes represent the DP distributions over outputs of another DP. Therefore, to conveniently represent the above process, we will take the view of this generative process as being a graph.

9.3 Nested Chinese Restaurant Processes[18]

The first model that we will talk about is the nested Chinese Restaurant Process(nCRP). nCRPs can be thought of in several ways. One is as a generalization of LDA. Consider the way in which documents are generated in LDA. A distribution of the topics is chosen for each document. Each word in the document is then generated by sampling a topic and subsequently the word from the distribution represented by that topic. Even in a nCRP, documents are generated through mixtures of a fixed number of topics. The only difference is that in the case of the nCRP, there is an infinite set of topics. So, how exactly does the nCRP choose topics for a document?

The topics in a nCRP are organized as the nodes of an infinitely branching tree of fixed height, L. Each node in the tree represents a topic. Each document now samples a path from the root of the tree to a leaf. The path now contains L nodes which are essentially topics. Each word now chooses a node along the path and generates a word. The hierarchical assumption in nCRPs is different from those made in the disambiguation model used here. For one thing, it is not necessary that children of a particular node have a topic anything similar to its parent. For example, since the root node is common to all the documents, one would expect that the salient words in the distribution for that topic are words common to all documents. This would lead us to give high priority to common words like function words or even stop words if they are not pruned during a preprocessing step. We will formalize this shortly.

Now, the question is how to model a distribution over the paths of an infinitely branching tree since this is how a document is generated. On

the one hand, we would like to provide documents with the opportunity of forging their own paths as this is one of the prime reasons for using nonparametric models in the first place. On the other hand, we would like to have several documents choosing similar paths so as to gain some knowledge about the distribution of topics in the corpus. Therefore, we would like to have a prior over such paths which provide both capabilities. The Dirichlet Process(DP) which we've seen before is well suited to this task.

A prior for the distribution over the paths of this tree is formed by using a Dirichlet Process(DP) at each node of the tree. The prior thus decomposes over the nodes of the tree. The tree will have a single root node and sampling from the DP at the root node will give the next node along the path. By repeating this process at every node till we sample L nodes from the root to the leaf of the tree, we can sample one path amongst all the root to leaf paths from the tree. Now, we just need to ensure that the prior satisfies our two requirements from the previous paragraph, that is new paths can be sampled if the data so desires it and also, documents are encouraged to choose nodes sampled before. Since we are using a DP at each node, given the tendency of the DP to discourage the choosing of new nodes as more and more new nodes are chosen, it will encourage documents to choose nodes chosen before which translates into choosing paths which have been chosen before. On the other hand, if a document is significantly different, it can overcome the influence of DP prior and still choose a new node which also allows it to choose a new path of itself. Therefore, using DPs to build a prior over the paths of the tree is a sensible choice.

We are now ready to describe the generative process associated with nCRPs. We will use, c_i to denote the i^{th} node in the path chosen by the document. To parameterize the DP at node, n_i, we will use γ_i to denote the propensity to sample a new node, m_i denotes the number of times node n_i has been sampled previously and n_{ij} denotes the number of times, node n_j has been selected from node n_i. The node represented by c_i will be n_{c_i}. θ will denote the distribution over topics for each document which is modelled as Dirichlet Distribution with parameter α. t_i will denote the topic for node n_i. η is used to parameterize the word distribution for each topic and β is used to represent the draws from η. The generative process for a document, d is as follows:

1. c_1 is chosen to be the root of the tree
2. For each level $l \in \{2, \ldots, L\}$:
   (a) Choose c_l according to the DP at c_{l-1}

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

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

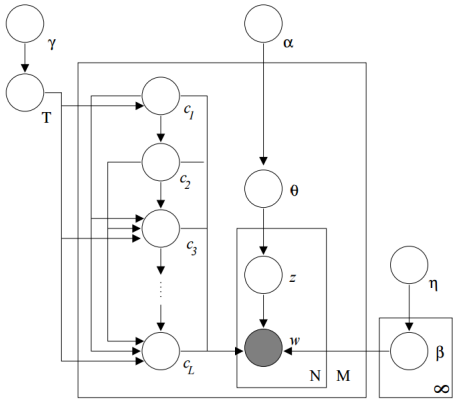


Figure 9.1: Plate Model of the Nested Chinese Restaurant Process

3. Now, that the path is selected a distribution over the, L topics is chosen as
$$\theta \sim Dir(\alpha)$$

4. For each word, w,
   (a) Sample, topic, $z \sim Mult(\theta)$
   (b) Sample $w \sim t_{c_z}$

A plate model of the nCRP is shown in figure 9.1. Intuitively, as explained before, we'd expect topics at the higher levels of the tree to be more general and common to a lot of documents in the corpus while the topics at the lower levels will be much more specific.

Some of the topic hierarchies learnt by using the nCRP model can be seen in 9.2. As we can see, the words at the root of the tree are much more general than the ones we see as we move down the tree. At the top, we have words like "the", "of" and "to" while at the next level, we see two distinct tracks of the NIPS conference where one is more focussed on Machine Learning and the other on Neuroscience with even more specificity as we move further towards the bottom of the tree. Thus, nCRPs can be used to learn topic hierarchies from corpora of text.

We will now discuss some of the problems with the nCRP model:

1. One problem is with the restriction that a single document can only sample a single path. This means that there is no scope for a document belonging to multiple paths and thus articles with a broad scope will not be well represented or will be forced to create a new path despite the fact that they can be well represented using some paths already discovered. For example, a NIPS paper about both Neuroscience and

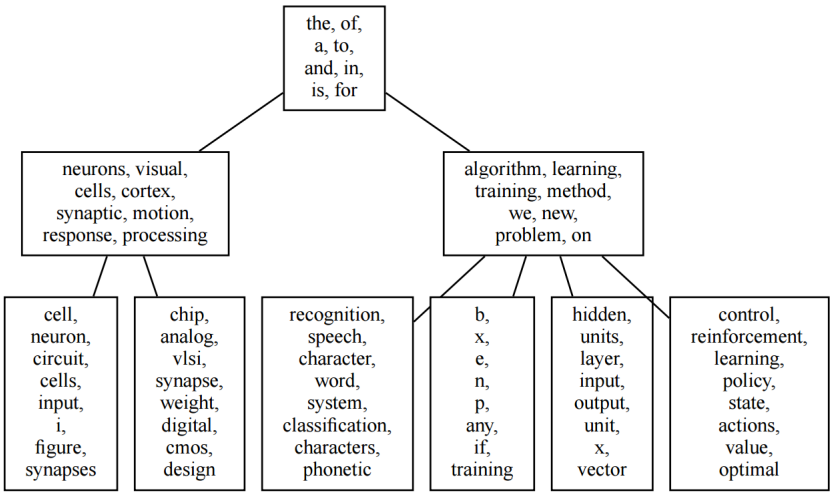


Figure 9.2: Results of Running the nCRP on NIPS Abstracts

Machine Learning will not be well represented in the nCRP model as it will either be forced into one of the two broad categories discovered or will create a new path despite the fact that the document bears several similarities to documents from either of the two classes and this similarity will not be captured by the nCRP model.

2. The other problem in the context of our task is that these may not be the type of topic hierarchies that we want to capture. In our framework, where we consider entities as topic, it does not make sense to have entities for extremely general topics like function words which the nCRP will capture. But, these topics may be useful to prune away extremely general words.

A solution to the first problem is to generalize the nCRP model to the nested HDP(nHDP) model which we'll discuss now while the second one will be looked at more closely in the subsequent chapter.

9.4 Nested Hierarchical Dirichlet Processes[19]

Nested Hierarchical Dirichlet Processes(nHDP) can be thought of a generalization of the nCRP in the same ways that HDPs are generalizations of the Dirichlet Process Mixture Models. The problems in the nCRP model arise because of the restriction that each document can only sample one fixed length path. nHDPs take away both of those restrictions. Each document no longer has to sample a single path. It also doesn't need to sample a fixed

number of topics(nodes). This is a much richer model which allows them to model complex documents which borrow from multiple different paths of the topic hierarchy. For example, instead of creating a new path for documents which talk about sports and medicine, a document can simply borrow elements from both paths.

This additional power that a nHDP enjoys comes with some complications. Since we no longer have a fixed finite number of topics to select from as was the case with the nCRP(L), there is a new problem of how to select a level for a particular word. In the nCRP, we simply drew a value from a multinomial distribution θ. This is no longer possible as we neither have trees of a fixed height nor are we dealing with single paths. We will now describe the nHDP model and how it solves the problem of sampling random nodes.

We start off with the nCRP model from before. In the nCRP model, each document draws a path from the root node to a certain height, L using DPs at each node of the infinite tree. In the nHDP, model, each document draws a distribution over nodes for each node in the tree. This is done, through the use of another DP at each node. This is similar to how the HDP generalizes the DPMM. Here, the DP model at the document level (this is identical to the nCRP model) is used a generate a distribution over the same nodes of the tree that are generated at the global level as part of the nCRP model except that this distribution is at the document level. There is one small problem here. Each word needs to sample a node in the tree. This was easy in the nCRP case. We had to sample a finite path and once the path has been sampled, one of the nodes on the path was selected as the topic for this word. In this situation, it is not so simple. We do not have a fixed path length. Therefore, what we actually have is a distribution over all finite paths in the tree. Furthermore, we need to sample a node from the tree.

In the current situation, we have a global nCRP model which we'll call τ. Once the document draws its own distribution over the same tree as the global nCRP model τ_d. Now, we need to somehow get a distribution over the nodes of the tree. This is done by sampling a Beta distribution with parameters $(1, \beta)$ distribution at each node. This will decide how much of the probability mass stays with the node and how much gets passed on to its children. Thus, the whole model can be looked at as the trickling down of probability mass from the root to the nodes in the tree. The root receives total probability mass 1 out of which it keeps a certain fraction (the draw from the Beta distribution) for itself and passes the rest to its children (according to the distribution τ_d). Now, we have a distribution over the nodes of the tree and thus a distribution over the topics in the tree for each node in the tree.

We will denote a node indexed by i as n_i and the topic corresponding to that node as t_i. G_i will be distribution over the children of the i^{th}

characterized by a CRP. Here is the generative process of the nHDP formally described:

1. Generate global nCRP τ according to the previous model without the restriction of L
   (a) This means generating a topic for each node, t_i
   (b) Also, generate a distribution over its children G_i using a DP
2. For each document, d
   (a) Generate a document specific tree τ_d
      i. The topics for the nodes remain the same
      ii. Draw new distributions over the children of each node, n_i as G_i^d using a DP
      iii. We now have a distribution over finite paths in the tree. This is guaranteed to be over the same nodes as the original tree but with different properties.
   (b) Generate for each node, n_i, a draw $B_i$from a Beta distribution with parameters $(1, \beta)$
   (c) For each word, w:
      i. Sample a node as follows, set $n_w = root$
         A. With probability B_{n_w}, sample n_w
         B. Otherwise, sample $B_{n_w} \sim G_{n_w}^d$ and repeat
      ii. Sample a word according to t_{n_w}

By allowing each document to sample from the whole tree and not just a single path, we allow documents to have complex mixtures of topics and alleviate problems that plagued nCRPs. Furthermore, by allowing the documents to choose the probability distributions over the nodes, the documents can choose their topic proportions but still share topics with all the other documents which allows the model to capture the broad topics of the corpus.

While the nHDP solves the first problem of the nCRP along with adding the ability to change the depth of the tree, it still doesn't solve the second "problem" that we noted. The types of topics that the model learns aren't topics that correspond to entities. Since our goal is entity disambiguation, we will need that our topics correspond well with the entities that we aim to disambiguate to. In the next chapter, we will present a model for disambiguation which also allows us to learn structure.

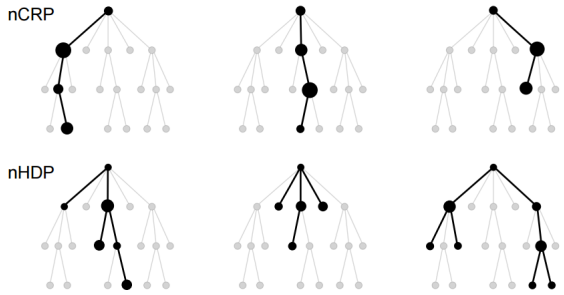


Figure 9.3: A qualitative comparison between nCRP and nHDP which show the different structures learnt by the two models for a single document

Chapter 10

Structure Learning for Disambiguation

In this chapter, we will look at how disambiguation algorithms can benefit by using models which can learn structure instead of just exploiting it. First we'll motivate why learning structure can be important for the types of disambiguation models discussed here. Then, we will propose a variant of the nHDP/nCRP scheme which can learn the type of structure that would work well for disambiguation systems.

10.1 Notation

In this chapter, we talk about a DP without explicitly defining its support. While this was easy to do in the models discussed previously where the support DP were over the set of possible topics, this is not so easy in the models discussed here. Here, we will have DPs whose support itself is the set of possible outcomes of a DP whose support is the set of possible topics. This can be looked upon as a tree where the output of one DP is a distribution over distributions over topics. Since, both distributions are over countable support, this gives the impression of an immensely large tree. This structure can be repeated as many times as required to get really large structures. We will avoid referring to such DPs as distributions over distributions . . . topics or in the case of nHDPs infinite distributions and rather look at them from the point of view of a graph where the nodes represent the DP distributions over outputs of another DP. Therefore, to conveniently represent the above process, we will take the view of this generative process as being a graph.

10.2 Why Learn Structure for Disambiguation Systems

The disambiguation system discussed earlier in assumes a fixed structure within which the topics are present. Topics can only be added to special category nodes but other than that, there is no flexibility to change the structure of the model being used. As discussed before, there are at least three reasons why we might want to learn this structure:

- Unavailability of a structure - We may not always be able to find a structured object. We may have found one here but that doesn't mean this will always be the case.
- Structure is insufficient - The structure provided may not be suitable to the task at hand.
- Structure needs to evolve - Often data sets grow over time, and this necessitates changing the structure to reflect the information that new data brings in.

The first reason is not relevant in the context of disambiguation systems unless one chooses not to use category graphs already available as part of Wikipedia or Yago. The next two reasons, however, can be very relevant to the disambiguation problem. The fact that structure needs to evolve is easy to see. Consider the Wikipedia page, `http://en.wikipedia.org/wiki/Harry_Potter`. This was a book released in 1997. Before that, no one, except the author, even know of its existence. Now, the book series has its own category. This is an example where the category structure of a Knowledge Base(in this case Wikipedia) changes with the introduction of a new entity/event. One can imagine that this kind of process happens on a regular basis with more and more information entering the internet. In situations like this, a static category structure may not fully take into account the evolution this new entity.

Even if the third problem is not seen as an important problem to solve, the second one quite relevant. Entity disambiguation systems assume that the category structure is relevant and exploit it for better accuracy. But the methods used are robust to noise in the category structure and the intra-Wiki links. Our model on the other hand is reliant on having a good category structure to work with and the assumption that the Wikipedia category structure if ideal for the task at hand might be faulty. Furthermore, the Wikipedia category structure is large with close to $900,000$ nodes. This makes inference difficult as sampling topics for new words becomes exceedingly difficult. Also, from the report in WPAM[16], we see that very few nodes(as few as 12) are actually relevant to the disambiguation process. Given this information, the $900,000$ nodes are redundant as well as

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
First-level_administrative_country_subdivisions	living_thing
Geography_of_Europe	organism
Counties_of_the_United_States_by_state	whole
Overpopulated_stub_categories	entertainer

Table 10.1: Salient Nodes for Wikipedia and Valid Nodes on YAGO2s for an Elizabeth Fretwell

noisy as we will see. Therefore, learning the category graph structure from data while imposing some constraints on its size may help improve disambiguation performance as well as speed up computation. We can see some examples of important valid nodes in Wikipedia near the top of hierarchy. We see that people and places are separated quite early on in the category structure. Also, shown are a list of nodes valid for an entertainer near the top of the hierarchy. Even here, we see that the entities that can occur close to the artist are mostly people. This is a problem as we are supposed to associate two thematically similar object that are used together a lot. For example, one would expect a guitarist to be associated with the brand of guitar he uses or a sportsperson with their sponsor, etc. Once categorization is done on such broad terms as people and places so close to the top of the hierarchy, there is the chance that the disambiguation system will suffer.

10.3 Wikipedia Structure Learning Model

We've seen previously that nCRPs and nHDPs both learn hierarchies of topics. The problem with these methods is that the types of hierarchies learnt(and subsequently topics) are not reflective of the entities in the document. In our setting, where entities are the topics themselves, it doesn't make sense that there is one entity that is always present in our document. Consider the topic at the root which consists of mostly functional words. This topic is not representative of any entity in the knowledge base as it contains words in every document. But on the other hand, the topics learnt are indicative of the theme of the documents at an individual level and not a broad classification of entities based on types like people or places. Exploiting these kinds of models can thus help disambiguation.

One simple way to incorporate one of these models into our disambiguation system is to simply run the respective system on the corpus. Once a topic hierarchy is obtained, we can attach the entities to the leaf nodes of the hierarchy based on a heuristic like where the documents they've occurred have mapped to most often, etc. Now, we run the algorithm described on the new hierarchy.

While the previous model may work well, we would like a model which simultaneously does both, learn structure and perform disambiguation. The earlier model might need to be revised often because the category structure evolves with more data being available. This might require rerunning the disambiguation system several times. A model which can do both simultaneously is much cleaner.

The set-up here is identical to the one where disambiguation takes place using a fixed category structure, we are given as input a corpus with some annotations. The goal here, however, is not just to perform disambiguation on unseen data but also learn a category structure that reflects the themes of the documents in the corpus. We will borrow from the nCRP/nHDP architectures in the previous section. We will describe here a model where we assume the category structure is a tree but simple extensions to model Directed Acyclic Graphs are possible.

We start off with the infinitely branching tree from the nCRP case with a bounded height, L. To avoid topics being about a general theme rather than individual entities, we allow topics only at the base of the tree and not all the way along the path. Also, in contrast to nCRPs, we will not have a single topic at the leaf, rather a distribution over topics which again are entities. We will borrow some of the ideas from the nHDP as well. Instead of sampling a path for each document, we will sample a distribution over the paths in a tree similar to the nHDP except that here, we will insist on sampling full paths. We will now describe a disambiguation system that also learns the structure of category graphs in addition to doing disambiguation.

The parameters of the model are γ, α, α_d, β, β_d, ω. The generative process of the document d is as follows:

1. At the global level:
   (a) Generate a global set of entities $G \sim DP(\omega, \gamma)$
   (b) Generate an infinite tree τ of height L using the nCRP model except without topics generated for the nodes (Using a DP with parameter β at each node)
   (c) For, each leaf, l of τ, associate a distribution over entities $G_l \sim DP(G, \alpha)$
2. At the document level, d is the document being generated:

(a) Generate an infinite tree τ_d of height L using the nCRP model except without topics generated for the nodes (Using a DP with parameter β_d at each node)

(b) For, each leaf, l of τ_d (Which coincide with the leaves for τ), associate a distribution over entities $G_l \sim DP(G, \alpha)$

(c) For, each word, w:

i. Sample a path to a leaf node, l_w according to the distribution in τ_d

ii. Generate a word according to the distribution at l_w which is G_{l_w}

10.4 Discussion About the Parameters of the Model

The parameters and the effects that they will have on the performance of the model can be summarized as follows:

1. ω - This is the base distribution over words for the Dirichlet Process generating the entities. Since, we would like each entity to have high probabilities only for a few words, we will prefer that ω gives high probability mass to word distributions which are sparse in their support.

2. γ - This is the propensity of the model to create new entities. High values of γ will encourage more new entities to be generated through the course of running of the model.

3. β - This will control the width of the graph. A high value of β will encourage more documents to take paths that haven't been sampled before in the case that the document specific value of the parameter β_d allows it. Wider graphs allow for greater variety in the documents that are produced. A high value of β will have a very detrimental impact as it may destroy the topic correlation that needs to be learnt through the model. Ideally, we'd like each document to be a part of a very thin part of the graph as we expect each document to be about only a small number of related topics. A high value of β makes this difficult to achieve as documents will now cover a large part of the tree and not learn topics that properly describe the corpus.

4. β_d - This parameter controls how much a word in the document will be allowed to sample a path that hasn't been taken by any of its fellow words. If it is set to a hight value, each document can sample from very diverse paths which can make the modelling of complex documents with multiple topics easier but on the other hand a value too high and topic correlation might be lost as it becomes very easy

to sample any entity with a topic. A value too low might force the model to associate entities with leaves they were never meant to be associated with because β_d is too stringent to allow it to sample other paths and therefore, the leaf has to compensate by allowing an entity it wasn't meant to.

5. α - This parameter controls the effective outreach of the leaves of the tree, τ. A high value of α will encourage more and more entities to become associated with the leaf. This can destroy topic correlation as it becomes very easy to add any entity to the leaf being considered. A value of α too low can also have dangerous side-effects. If it does not allow new entities to evolve, the new entity will be absorbed into the topic for an entity currently at this leaf in which case we lose out on the disambiguation of an entity.

6. α_d - This parameter controls how likely a leaf will be to accept new entities on a document level. We would want to set this parameter lower than α because we expect more entities to be associated with the leaf on a global level rather than on a local level. A high value here will encourage more entities from the leaf to become associated with the document. This could hurt the word correlation for the other entities in the document if

10.5 Why will this model work?

The last question that needs answering is why this model will work. The nature of the model is such that it encourages most words to choose paths which are similar to the other words in the document. This means that the topic assigned to a certain leaf is likely to occur with more topics which are very similar to it. This is because of the fact that topics which are similar in theme tend to occur together in the same documents. When a document is being used for training, the topics in the document will prefer paths which share a long prefix as this maximizes the likelihood as far as paths to leaves are concerned. The model will actively discourage paths which diverge too much from each other. Even if some noise was encountered, that topic will be pushed out by the influence of the rest of the topics which will occur far more frequently than the noisy topic in the same group. Thus, the model encourages the topics in a document to go to close by parts of the tree. This means that the entities relating to a particular theme will tend to collect in the same part of the tree and this discourages other entities from coming to this part of the tree. This will cause thematically similar entities to collect in the same part of the graph thus establishing Topic Correlation and Word Correlation happens due to the association of the labelled topics to the documents.

10.6 Extensions

The one thing not handled by this model is modelling of entities with multiple themes. In the current model, if an entity has to be relevant for two themes, it needs to be associated with leaves in both parts of the graph Ex: Neuroscience is of interest both to Biologists and Computer Scientist and is an active area of research on both fronts. On the other hand, if we had a model for DAGs, then we could have Neuroscience deriving from both Computer Science. A simple extension to the model will allow it to model DAGs as well. This can be done by imposing a third level Dirichlet Process prior on the nodes at each level of the graph. This will force the edges from different parts of the DAG on the same level to go to the same set of nodes at the next level thus forming a DAG. The discretization of the nodes in the graph allow different nodes on the same level share the same set of nodes on the next level. Further generalizations allow the full modelling of a DAG by allowing edges across different heights as well.

Bibliography

[1] Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. "A Framework for Benchmarking Entity-Annotation Systems". In: *Proceedings of the International World Wide Web Conference (WWW) (Practice & Experience Track)*. 2013.

[2] Stephen Dill et al. "SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation". In: *Proceedings of the 12th international conference on World Wide Web*. ACM. 2003, pp. 178–186.

[3] Rada Mihalcea and Andras Csomai. "Wikify!: linking documents to encyclopedic knowledge". In: *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. ACM. 2007, pp. 233–242.

[4] Razvan C Bunescu and Marius Pasca. "Using Encyclopedic Knowledge for Named entity Disambiguation." In: *EACL*. Vol. 6. 2006, pp. 9–16.

[5] R. Guha and Rob McCool. "TAP: A Semantic Web Platform". In: *Comput. Netw.* 42.5 (Aug. 2003), pp. 557–577. ISSN: 1389-1286. DOI: 10.1016/S1389-1286(03)00225-1. URL: http://dx.doi.org/10.1016/S1389-1286(03)00225-1.

[6] Silviu Cucerzan. "Large-Scale Named Entity Disambiguation Based on Wikipedia Data." In: *EMNLP-CoNLL*. Vol. 7. Citeseer. 2007, pp. 708–716.

[7] David Milne and Ian H Witten. "Learning to link with wikipedia". In: *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM. 2008, pp. 509–518.

[8] Lev Ratinov et al. "Local and global algorithms for disambiguation to wikipedia". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics. 2011, pp. 1375–1384.

[9] Sayali Kulkarni et al. “Collective annotation of Wikipedia entities in web text”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2009, pp. 457–466.

[10] Xianpei Han, Le Sun, and Jun Zhao. “Collective entity linking in web text: a graph-based method”. In: *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM. 2011, pp. 765–774.

[11] David J Aldous. *Exchangeability and related topics*. Springer, 1985.

[12] Wikipedia. *Parametric model — Wikipedia, The Free Encyclopedia*. [Online; accessed 13-May-2015]. 2015. URL: `http://en.wikipedia.org/w/index.php?title=Parametric_model&oldid=661850354`.

[13] David M Blei, Andrew Y Ng, and Michael I Jordan. “Latent dirichlet allocation”. In: *the Journal of machine Learning research* 3 (2003), pp. 993–1022.

[14] Yee Whye Teh. “Dirichlet process”. In: *Encyclopedia of machine learning*. Springer US, 2010, pp. 280–287.

[15] Yee Whye Teh et al. “Hierarchical dirichlet processes”. In: *Journal of the american statistical association* 101.476 (2006).

[16] Saurabh S Kataria et al. “Entity disambiguation with hierarchical topic models”. In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2011, pp. 1037–1045.

[17] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. “Yago: a core of semantic knowledge”. In: *Proceedings of the 16th international conference on World Wide Web*. ACM. 2007, pp. 697–706.

[18] DMBTL Griffiths and MIJJB Tenenbaum. “Hierarchical topic models and the nested Chinese restaurant process”. In: *Advances in neural information processing systems* 16 (2004), p. 17.

[19] John Paisley et al. “Nested hierarchical Dirichlet processes”. In: (2012).