Structured Entity Identification and Document Categorization: Two Tasks with One Joint Model

Indrajit Bhattacharya
IBM India Research Lab,
New Delhi
indrajbh@in.ibm.com

Shantanu Godbole
IBM India Research Lab,
New Delhi
shgodbol@in.ibm.com

Sachindra Joshi
IBM India Research Lab,
New Delhi
jsachind@in.ibm.com

ABSTRACT

Traditionally, research in identifying structured entities in documents has proceeded independently of document categorization research. In this paper, we observe that these two tasks have much to gain from each other. Apart from direct references to entities in a database, such as names of person entities, documents often also contain words that are correlated with discriminative entity attributes, such age-group and income-level of persons. This happens naturally in many enterprise domains such as CRM, Banking, etc. Then, entity identification, which is typically vulnerable against noise and incompleteness in direct references to entities in documents, can benefit from document categorization with respect to such attributes. In return, entity identification enables documents to be categorized according to different label-sets arising from entity attributes without requiring any supervision. In this paper, we propose a probabilistic generative model for joint entity identification and document categorization. We show how the parameters of the model can be estimated using an EM algorithm in an unsupervised fashion. Using extensive experiments over real and semi-synthetic data, we demonstrate that the two tasks can benefit immensely from each other when performed jointly using the proposed model.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining; I.5.1 [**Pattern Recognition**]: Models—*Statistical*; I.5.4 [**Pattern Recognition**]: Applications—*Text processing*

General Terms

Algorithms, Theory

Keywords

document categorization, entity identification, probabilistic generative model

KDD'08, August 24–27, 2008, Las Vegas, Nevada, USA.

1. INTRODUCTION

The importance of unstructured data as an information source is increasing steadily, both in the scientific and business communities. Recent Gartner [1] research reports suggest that 80% of data in today's enterprises is unstructured. In the growing competitive landscape, enterprises cannot afford to let such huge amounts of data and insights hidden therein go untapped. There has also been a lot of activity in the research community towards deriving insight from unstructured documents and integrating heterogeneous data sources. In this paper, we focus on two tasks in unstructured data — entity identification and document categorization — that are of critical importance, but have traditionally been investigated independently.

Entity identification in unstructured data forms a critical component of data integration, which has attracted significant amount of research in recent years[7, 5]. Given a structured database containing descriptions of various entities, and a collection of unstructured documents mentioning or referring to one or more entities from this database, entity identification attempts to identify the most relevant database entity (or set of entities) for a given document. The task is often made challenging when the mentioned attributes do not exactly match the entity descriptions in the database due to misspellings and other kinds of noise, or when the evidence available in the document is insufficient for pinning down any single entity.

To illustrate, consider a movie database such as IMDB [2] and movie reviews written by various users. Figure 1 shows sample movies from the database. The database contains movies with information such as movie name, actors, directors, writers, genre of the movie, and its rating, among myriad other information. Figure 2 shows some snippets of reviews written by different online users for movies from this database.

In this scenario, the goal of entity identification is to tag movie reviews with the most appropriate movie from the database. This is typically achieved by first annotating named entities in the documents and then identifying entities based on these annotations. The annotations in our example reviews are highlighted in Figure 2. For example, review (a) has two annotations, 'Harrison Ford' and 'David Twohy'. These annotations uniquely map the review to the fourth movie entity, *Fugitive*, in the database. However, for the remaining reviews, the annotations do not have enough information to unambiguously identify a unique movie.

[1] `http://www.gartner.com`
[2] `http://www.imdb.com`

	Movie Name	Actor	Writer	Genre	Rating
1.	Indiana Jones and Last Crusade	Harrison Ford	George Lucas	Adventure	Excellent
2.	American Graffiti	Harrison Ford	George Lucas	Comedy	Average
3.	Star Wars: Return of the Jedi	Harrison Ford	George Lucas	Sci/Fi	Excellent
4.	Fugitive	Harrison Ford	David Twohy	Action	Good

Figure 1: Sample Movies from IMDB

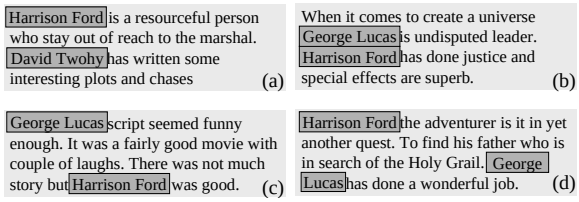


Figure 2: Snippets of Movie Reviews from IMDB

An equally important and well-researched task over unstructured document collections is that of document categorization[11, 16]. This involves automatically tagging documents in a collection with labels from a pre-defined label set. In our movie example, it is desirable to automatically organize all reviews according to the genres for easy browsing and retrieval. Note that it may be equally useful and meaningful to organize the same collection of reviews according to some other label-set, for example, the popularity of the movies or the demographics of the writer. In the classification setting, a critical requirement is sufficient and accurate supervision, which is typically provided in the form of training samples for the different labels. However, designing meaningful label-sets and building training data is time consuming, error prone and highly human-intensive. In the document categorization setting, we investigate if this need for supervision can be alleviated.

In this paper, we consider these two text mining tasks together and motivate how they can mutually reinforce each other under certain scenarios. In our example, entity identification based solely on the annotated terms in the document often fails to unambiguously identify an entity. We observe that the unstructured words in the document that are not annotated, and are typically thrown away during the entity identification process, often carry useful information. Although they do not directly mention an entity attribute, they may provide significant *indirect* evidence for *inferring* database column values. As an example, for the review in Figure 2(b), words such as *adventurer*, *quest* and *search* are suggestive of a movie from the 'Adventure' genre. When combined with the evidence from the annotated terms, it clearly points to the first movie, *Indiana Jones and the Last Crusade*. In enterprise settings, such as Banking and CRM, the words used in a customer communication are often clearly indicative of her age-group and income level. However, discriminating these categories from words in the document requires that we have trained classifiers based on interesting database column values.

Now, consider the document categorization side of the story. We have seen that supervision in the form of labeled training samples is a critical requirement for effective categorization. However, if we are able to automatically and accurately identify entities for documents, then the database columns values provide labels for the documents. In our example, if all the reviews are correctly linked to their corresponding movies, then review (a) would be labeled as 'Action', review (c) as 'Comedy' and so on. Then, by aggregate analysis, any classifier would discover that words such as *adventurer* and *quest* are indicative of 'Action' movies, while *funny* and *laugh* are suggestive of the 'Comedy' genre. In the CRM setting, it is possible to discover vocabularies used by different categories of customers.

Thus, quite clearly, the two tasks can be combined to their mutual benefit. In this paper, we explore how the two tasks can be formally related, and propose a probabilistic model for jointly performing entity identification and document type categorization.

Our contributions: We motivate the problem of jointly addressing the two tasks of entity identification and document categorization and show how they can naturally benefit each other in many domains. We propose a probabilistic generative model for documents that accounts for unstructured words and mentions of entity attributes, and inference in this model jointly addresses the two tasks. We show how parameters can be estimated in this model in a completely unsupervised fashion from a document collection and a backend structured database. We demonstrate using experiments over real and semi-synthetic data that our model improves entity identification over state-of-the-art baselines and enables unsupervised document categorization to compete with supervised approaches.

Outline: Next, we formulate the joint problem and explore ways to relate the two tasks in Section 2. Then, we describe our joint probabilistic model in Section 3 and the unsupervised parameter estimation algorithm for it in Section 4. We describe experimental results over different datasets in Section 5. We discuss our contributions in the light of related research in Section 6 and finally conclude in Section 7.

2. PROBLEM FORMULATION

In this section, we formulate the technical problem of jointly addressing entity identification and document categorization. We start by formulating the two tasks independently, and then motivate why and how they can be addressed jointly.

Entity Identification: The first problem is *entity identification*, where we are given a structured database that contains a relation with k columns $\mathcal{C} = \{C_i\}$. We will refer to each row of the relation as an entity e, having its own values $e.C_i$ for the k columns. We also have a collection $\mathcal{D} = \{d_i\}$ of documents, such that each document is about a specific entity from the database. More formally, each document d contains a set of attribute terms $d.A = \{a_i\}$. If e_i is the central entity for this document, then each attribute term a_i corresponds to some column value $e_i.C_j$ of entity e_i. For instance, in our previous IMDB movie database example, the columns in the relation are Actor1, Director1, Genre, etc. Each row in the database corresponds to a movie entity which has its own values for these columns. Our document collection consists of movie reviews, each of which is about a specific movie from the database and mentions its actors, actresses, director, among other things. These tokens form the attribute terms of the document. Given this bag-of-terms, our goal is to identify the central entity from the database

relation. In our example, the goal is to identify the movie from the review assuming enough evidence is present for the inference. The major challenge is that explicit identifiers of the entity, such as a unique movie id, or sometimes even movie name, may not be available in reviews. This holds true especially for noisy documents where even a_i may not exactly match its corresponding column value of $e.C_j$. In our example, the database contains fully qualified names of people which are unlikely to be used in a review. Just 'George Lucas' is ambiguous in the IMDB database and the full name of the director of our interest is 'George Walton Lucas Jr.'. Misspellings and abbreviations are also common, all adversely affecting recall. Precision can also be affected by noisy and partial information in the document if it leads to an incorrect entity being identified.

Document categorization: A critical corpus-level task that needs to be performed over the document collection is *document categorization*. Let $\mathcal{W}$ be the vocabulary of unstructured words that are used in the document collection. Each document d_i in the collection has its own set of words $d_i.W = \{w_{i1}, w_{i2} \ldots\}$ drawn from the vocabulary. Categorization groups documents in the corpus into different classes or categories based on the words they contain. This is helpful in organizing the document collection and making it amenable for navigation, browsing, and retrieval.

In the clustering formulation, the categorization process is unsupervised, but categories are not pre-specified. Instead, documents are grouped according to how similar/dissimilar they are in terms the words they contain using projections in some vector space model. In contrast, the classification approach supervises how the documents should be categorized by specifying a label-set and providing examples of documents for each label. An interesting and important aspect of classification is its multi-faceted nature. The same document collection can be categorized differently by providing a different set of labels. For example, the same collection of movie reviews can be classified equally meaningfully according to genre, rating, budget, country of origin, awards etc. This, however, comes at the cost of having to provide sufficient amount of labeled training data for each facet (label-set), which is costly, time-consuming, and often infeasible for some applications. Clustering, on the other hand, comes for free in terms of providing supervision, but can only produce a single facet based on the dominant features of the collection for a given similarity measure. For example, if the best grouping of the review collection is based on the movie ratings, then that is the separation that clustering would yield.

Relating the two tasks: In many scenarios, categorization and entity identification can be performed jointly and can benefit from one another. Imagine documents containing both attribute terms $d_i.A$ mentioning entities and unstructured words $d_i.W$, such that the words are strongly correlated with the column values of the central entity. This happens naturally in many real-world domains. As motivated in the introduction, the words used in a movie review are different for movies of different genres, as also for movies of different popularity ratings. In the enterprise CRM setting, different customers may complain differently even about the same products, depending on their demographic background.

When such correlations exist, both entity identification and document categorization can benefit. If some column values of the entity can be predicted from the words contained in the document, then these words provide additional evidence for identifying the entity. Additionally, since the column value is inferred using aggregate statistics, it provides robustness against noise. For categorization, it opens up the possibility of getting the best of both the classification and clustering worlds. If structured entities from the database can be associated with documents, then the correlated column values can act as multiple label sets for these documents. This 'supervision' can be used to perform multi-faceted categorization over the corpus without explicitly using labeled training data.

We next explore possible ways to associate the vocabulary and entities in the database and enable their mutual reinforcement. Observe that associating words with specific entities is not helpful in any direction. It does not help entity identification unless the words are known for the specific entity. Nor does it help document categorization as the size of the label set would be huge.

Relating words and entities: Instead of associating words with specific entities, we relate words to groups of similar entities in the database. We describe two natural ways of doing this. The first scenario is the *relation-based-vocabulary* scenario where the entities are partitioned over multiple relations $\mathcal{R} = \{R_i\}$, with directly corresponding columns. We have different word distributions associated with different relations R_i. For movies, imagine that we have organized (English) movies under different relations according to their country of origin. We have words associated with each country (or, equivalently each relation) and the words $d_i.W$ in a movie review d_i are picked from the vocabulary of the relation containing the movie. The second scenario is the *column-value-based-vocabulary* scenario, where all entities belong to the same relation R. This relation has special columns $\mathcal{T} = \{T_i\}$ which are called the "type columns", and the words in a document depend on the values $\{e.T_i\}$ in the type columns for the central entity e_i from R.

Equivalence of the formulations: The *relation-based-vocabulary* scenario can easily be mapped to the *column-value-based-vocabulary* scenario. In the movie domain, we can add all the movies from the different relations in $\mathcal{R}$ into a single relation R with a new type column T^* which indicates the original relation $R_i \in \mathcal{R}$ for the movie entity. The words in a movie review are determined by the value $e_i.T^*$ for the movie e_i. This reduction uses a single entity type column, but we can show that the multi-column setting is also not any more expressive. We can reduce it to the relation-based-vocabulary scenario by creating one relation R_i in $\mathcal{R}$ for every combination of values in the type columns. We can then populate these relations with entities that have that specific (entity, type value) combination. The relation-based and column-value-based scenarios are thus equally expressive; the single-column model is also as expressive as the multi-column model.

For the rest of this paper, we will focus on the single-column-value-based-vocabulary scenario, without any loss in generality. Specifically, we have all entities in a single relation R with k columns C_i. In addition to these k columns, we have one "type column" T containing values $\{t_i\}$. Each value t_i for T has different words from the vocabulary $\mathcal{W}$ associated with it. Each document d_i in the collection $\mathcal{D}$ is about a central entity e_i. It contains attribute terms $d_i.A$

derived from the column values $e_i.C_j$ of the central entity. Additionally, the document contains words $d_i.W$ from the unstructured vocabulary $\mathcal{W}$ that are associated with the type value $e_i.T$ of the central entity. Our goal is then to identify the central entity from the document d_i and also categorize the documents according to type values t_i. The challenge is that no labeled data is provided for learning the word distributions for the type values.

3. A JOINT PROBABILISTIC MODEL

In this section, we propose a joint probabilistic model that unifies entity identification and document categorization by relating word distributions with entities in the structured database. The probabilistic model is shown in the plate representation in Figure 3. It encapsulates three plates, or repetitive processes. The outermost plate describes the generation process for each document. This repeats N times, once for each document in the collection. Among its parameters, the model has a prior probability $P(T = t)$ over possible values for the type column T. The generation process for each document first chooses a value t for the type column T using the distribution $P(T)$. This chosen value t for the type column plays the key role in relating the attribute terms and the unstructured words in the document.

On the unstructured side, this type value determines the words from the unstructured vocabulary $\mathcal{W}$ that are chosen for the document. This is done using a distribution $P(W|T = t)$ over words $w \in \mathcal{W}$ for each value t of the type column. The smaller plate on the right shows the generation process for the unstructured words in the document. The process repeats m times, once for each word in the document. In each iteration, a word is chosen from W using the distribution $P(W|T = t)$ for the chosen type column value.

Let us now see how the attribute terms are chosen. For each type value t, the model has a distribution $P(e|T = t)$ over entities e from the database that have value t for their type column. The model first chooses an entity e using the distribution $P(E = e|T = t)$. This is the central entity e in the document. Additionally, there is a prior distribution $P(C = c)$ over the k columns of the relation. These two distributions allow us to generate the attribute terms a in the document. This process in captured using the smaller plate on the left. It repeats n times, once for each attribute term a in the document. In each iteration, the model chooses a column c from the prior distribution $P(C = c)$ over columns. This determines the specific entity column $e.C$ to be mentioned as the attribute term. Then the attribute term a is generated using a noisy process from the column value $e.C$ of the chosen entity e.

In summary, the model parameters Θ include the following distributions — $P(C)$ over columns in the relation, $P(T)$ over values for the type column, $P(E|T)$ over entities in the database and $P(W|T)$ over words in the unstructured vocabulary for each value t of the type column. The observed variables in the model are the unstructured words $w \in d.W$ and the attribute terms $a \in d.A$ contained in the document. They are shown shaded in Figure 3. The entity e, the type value t and the columns c for the attribute terms are all hidden and need to be inferred from the observed variables.

The probability $P(a|E = e, C = c)$ describes the noise process that generates the attribute terms based on the column values of the entity in the database. While it is possible to incorporate this as a parameter in the model, this has certain disadvantages. This needs to capture all different ways in which noise can be introduced in the domain, and explicitly model all of them. This would tailor the model too closely to specific domains, and make it hard to generalize. Enumerating and assigning probabilities to all different ways for transforming an entity column value also is an extremely laborious process. Instead, we can make use of the fact that the document collection has already been provided to us. So, we can safely make a closed world assumption, and for any column value in the database, consider only those transformations for it that occur in the documents in the collection. For example, for the name 'Harrison Ford', the transformations that need to be considered for it are only 'Ford', 'Harrison Ford', 'H. Ford' and 'Harrison', if those are the only 'similar' names that occur in the collection. Though 'Harry Ford' can be a valid candidate in general, we do not need to assign a probability to it, if it never occurs in the reviews that we have in hand. The issue here is in determining which terms in the document are 'similar'. We handle this by considering as input a similarity measure for each column $c \in \mathcal{C}$ in the relation. Then, given any column value from the database, the similarity measure specified for that column automatically defines a distribution over all attribute terms in the document collection. As an additional advantage, specifying the similarity measures decouples the model from the underlying domain.

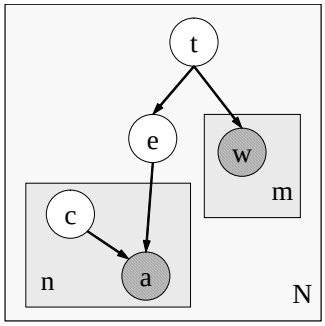


Figure 3: Probabilistic generative model for documents with attribute terms and unstructured words

We will now quickly run over the generation process to see how movie reviews are generated by our model in the example movie domain. Here, the type column corresponds to genres of movies, so the model has a prior distribution over movie genres. So first, a genre, say 'Action', is picked using the distribution $P(Genre = x)$. For each genre, the model has two distributions, $P(Movie = m|Genre)$ over movies , and $P(Word = w|Genre)$ over unstructured words. Once the genre is chosen, the unstructured words (such as *adventurer*, *quest*, *justice*, etc.) are generated for the document using the word distribution for the selected genre 'Action'. On the structured side, a movie, say "Raiders of the Lost Ark", is chosen from the 'Action' genre using the distribution over 'Action' movies. Finally, attribute terms are generated for this movie. This is done by repeatedly choosing a column from the movie table, such as 'Actor', 'Director', 'Producer', 'Production House', etc. using a prior distribution over movie columns, and then generating a noisy form of the chosen column for the movie. For example, for "Raiders of the Lost Ark", we may first choose 'Actor', get to "Harrison Ford" and generate "Ford" as the attribute term, then

choose 'Director', select "George Lucas" and generate "Lucas" as the second attribute term, and so on.

Consider a document d_i containing m unstructured words $w_i = \{w_{ij}\}$ and n attribute terms $a_i = \{a_{ij}\}$. The hidden variables for this document are the entity label e_i, the type value t_i and the columns labels $c_i = \{c_{ij}\}, j = 1..n$ for each attribute term. The joint probability distribution over these variables given the model parameter Θ is defined as:

$$\begin{aligned} & P(d_i|\Theta) \\ = \; & P(t_i, e_i, c_i, a_i, w_i|\Theta) \\ = \; & P(t_i)P(e_i|t_i)P(c_i)P(a_i|e_i, c_i)P(w_i|t_i) \qquad (1) \\ = \; & P(t_i)P(e_i|t_i)\prod_{j=1}^{n} P(c_{ij})P(a_{ij}|e_i, c_{ij})\prod_{j=1}^{m} P(w_{ij}|t_i) \\ = \; & P(t_i)P(e_i|t_i)\prod_{c} P(c)^{n_i^c}\prod_{j=1}^{n} P(a_{ij}|e_i, c_{ij})\prod_{w} P(w|t_i)^{n_i^w} \end{aligned}$$

where we have factorized the joint distribution according to the independence assumptions in the model, and n_i^c and n_i^w are the number of times column label c and word w respectively occur in the document d_i.

If the document collection $\mathcal{D}$ contains N documents $\{d_i\}$, then, assuming that each document is generated independently of the others, the likelihood of the document collection given the model parameters Θ can be given as $P(D|\Theta) = \prod_{i=1}^{N} P(d_i|\Theta)$.

Formalizing the intuition: The factored joint distribution in Eq. (1) helps to explain the intuition behind how the two tasks of entity identification and document categorization reinforce each other. Traditional entity identification considers $P(c_i)P(a_i|e_i, c_i)$, possibly augmented with a prior $P(e_i)$ over the entities, for choosing the most likely entity for the document. In our model, the words in a document come into play to determine the 'type' for the document, and entities that belong to that particular type are more likely to be right entity for the document. This helps to reduce the ambiguity in identifying entities. On the document categorization side, standard document categorization would consider $P(t_i)P(w_i|t_i)$ for choosing the most likely type for the document. But, in Eq. (1), the attribute terms in the document play a role in identifying entity candidates for the document, and only the entities identified for the document contribute relevant type labels for it. We will see in Section 5 that the experimental results back up this intuition.

4. PARAMETER ESTIMATION USING EM

In this section, we discuss how the model parameters, described in Section 3 are estimated from a given document collection and the algorithmic challenges that are involved. Note that we do not require labeled training data for estimating the model parameters, and instead resort to unsupervised estimation using the Expectation Maximization approach. Using standard EM derivation and incorporating the sum constraints for each probability distribution, the update rules for the model parameters look as follows:

$$P(t_i = x) = \sum_{i=1}^{N} P(t_i = x|a_i, w_i)/N$$

```
      Algorithm LearnModel (Doc Set 𝒟, Relation ℛ)

1.    Initialize model parameters
2.    Iterate until convergence or k times
          % E step
3.        for each document d in 𝒟
4.            for each topic t in T
5.                for each entity e in ℰ
6.                    for each col assign c to d.A
7.                        compute P(t, e, c|d)
8.                    compute P(t, e|d)
9.                compute P(t|d)
10.           for each column c in 𝒞
11                compute P(c|d)
          % M step
12.       for each topic t in T
13.           re-estimate P(t)
14.           for each entity e in ℰ
15.               re-estimate P(e|t)
16.           for each word w in 𝒲
17.               re-estimate P(w|t)
18.       for each column c in 𝒞
19.           re-estimate P(c)
```

Figure 4: The basic EM algorithm

$$P(w|t_i = x) = \frac{\sum_{i=1}^{N} n(i, w)P(t_i = x|a_i, w_i)}{\sum_{w \in \mathcal{W}} \sum_{i=1}^{N} n(i, w)P(t_i = x|a_i, w_i)}$$

$$P(e_i|t_i) = \frac{\sum_{i=1}^{N} P(t_i, e_i|a_i, w_i)}{\sum_{e \in \mathcal{E}} \sum_{i=1}^{N} P(t_i, e_i|a_i, w_i)}$$

$$P(c) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{n} P(c_{ij} = c|a_i, w)}{\sum_{c \in \mathcal{C}} \sum_{i=1}^{N} \sum_{j=1}^{n} P(c_{ij} = c|a_i, w)}$$

In order to do these updates, we need the posterior probabilities of the hidden variables computed for each document. They can be computed in a straight forward manner by marginalizing over the joint distribution:

$$P(t_i, e_i, c_i|a_i, w_i, \Theta) = P(t_i, e_i, c_i, a_i, w_i|\Theta)/P(a_i, w_i)$$

where $P(a_i, w_i) = \sum_t \sum_e \sum_c P(t_i = t, e_i = e, c_i = c, a_i, w_i)$

$$P(t_i|a_i, w_i) = \sum_e \sum_c P(t_i, e_i = e, c_i = c|a_i, w_i)$$

$$P(c_i|a_i, w_i) = \sum_t \sum_e P(t_i = t, e_i = e, c_i|a_i, w_i)$$

The basic EM algorithm is described using high-level pseudo code in Figure 4. However, it poses several scalability and other challenges which we describe and resolve next.

Scalability Issues: We first consider the scalability issues that arise in the E-step of the algorithm. This is clearly brought out by lines 3-9 in Figure 4. Consider a document d that has n attribute terms. Then this document has $n+2$ latent variables — t for the type, e for the entity and a c variable for each of the n attribute terms, indicating the column that it corresponds to. The unrestricted space of assignments to these $n+2$ variables is clearly intractable. If there are E entities in the database, T possible values for the type column, k different columns, then there are $E \times T \times k^n$ possible value assignments to the $n+2$ hidden variables for each document. Fortunately, very few of these assignments have non-zero posterior probability for the document. So our aim is to identify a significantly smaller space of assignments per document that need to be evaluated.

First, observe that the exploring the entire $E \times T$ space is clearly not necessary. We consider domains where each entity has only one value for the type column. This drastically reduces the number of assignments for the entity and type values to E from $E \times T$.

We next focus on restricting the number of entities that we need to consider for any document d. For each attribute term a_i in the document d, we identify the candidate matches for this term across the various columns of the database using the similarity measure that has been specified for this attribute. Each candidate match for a term a_i consists of an entity index e_i, a column index c_i and a score s_i. For example, the attribute term 'Johnny Gray' in a movie review returns $(director, 7395, 0.789)$ as a match, indicating that it matches the director column of movie 7395 (which has value "Gray, John III" in the database) with score 0.789. Note that the score is determined by the similarity measure that is employed for that attribute. The matches for all the terms in a document allow us to identify what assignments to the hidden entity and column values are relevant for this document. We extract the set of all entities that occur in a candidate match for at least one term in the document. These form the set of candidate entities $eCand(d)$ for the document.

Now, we look at the column assignment space for attribute terms. For each entity e in $eCand(d)$, clearly not all attribute terms in document d can map to all columns for e. From the candidate matches for each term a_i, we find the columns that are matched by this term for entity e. Potentially, there can be multiple such columns for the same entity. The term 'Bergman' in a review matches the director 'Ingmar Bergman', the writer 'Ingmar Bergman' and the actress 'Ingrid Bergman' columns for the Swedish movie 'Höstsonaten'. But barring such few and exceptional cases, a term matches at most one column for any entity. So if we get a maximum of e^{max} entities in the candidate set $eCand(d)$ for any document, then we have reduced $E \times T \times k^n$ possible assignments to a maximum of e^{max} assignments to evaluate. Note that e^{max} is a significantly smaller number than E, and grows extremely slowly with increasing number of entities in the database.

Initialization: Initialization is a critical issue to avoid local maxima in EM. While initializing all probabilities uniformly is an option, it usually does not work very well. We make use of the candidate matches for attribute terms and their corresponding scores for model initialization. Instead of first initializing the parameters, we first initialize the posteriors using the candidate match scores, and then take estimates of the parameters from the initial posteriors. For each entity e in $eCand(d)$ described above, we compute its score for the document d by summing over its scores from each attribute term a in the document: $score(e, d) = \sum_{a \in d.A} \max_c score(e, a, c)$. The candidate matches provide us with $score(e, a, c)$, and for each attribute term, we take the maximum contribution towards each entity over all its column matches. Then the initial entity posteriors are computed by normalization:

$$P_{init}(e|d) = score(e, d) / \sum_{e \in eCand(d)} score(e, d) \qquad (2)$$

These initial estimates do not consider the words in the document, and as such, serve as a baseline for comparison with our joint model. We score the columns for each document using the candidate matches in a similar way, and them normalize them to compute the column marginals $P_{init}(c|d)$. To compute the marginals over type values, we uniformly distribute the posterior for each entity over all its corresponding type values, and then appropriately normalize to compute $P_{init}(t|d)$ and $P_{init}(t, e|d)$. For the words, we score words and type values via the documents that contain the words, as $score(w, t, d) = P(t|d)n(w, d)$, where $n(d, w)$ is the number of occurrences of word w in document d. Normalizing the scores provides initial estimates for $P_{init}(w, t|d)$. Starting from these estimates for the document posteriors, we get initial estimates for the model parameters.

5. EXPERIMENTAL EVALUATION

In this section, we evaluate entity identification and document categorization performance of our model over different datasets and compare against state-of-the art baselines for both tasks. We have developed this model in a research engagement in a Customer Relationship Management (CRM) setting in a leading financial institution. Our huge unstructured document collection consisted of customer emails and agent call logs (summaries of interactions between customers and agents) from the financial institution's contact (call) center. The back-end structured data warehouse of customers, accounts and transaction details spans over 10 relations and 150 columns. We addressed the task of identifying the customer entity for each call log or email, and also categorizing the documents according to customer demographics, and the type of financial instrument such as credit card, debit card, savings or salary account used by them. We were able to achieve very good performance over both tasks inspite of very noisy textual data. But, owing to the confidential nature of this data, we are unable to publish the experimental details for it. As a shareable and repeatable alternative, we consider the IMDB dataset and focus on the tasks of identifying movies from reviews and categorizing reviews according to the movie genre. Additionally, to gain further insight into the performance of our model, and investigate how our results can generalize to datasets from other domains, we perform experiments on semi-synthetic data, where we can control data characteristics that influence performance of our model. Processed versions of the IMDB dataset used in our experiments are made available for download[3] as per repeatability guidelines.

5.1 Compared Approaches

We now describe the various baselines for the entity identification and document categorization tasks, and their parameters.

Document Classification Baseline: We use a state-of-the-art document classification baseline that makes use of *all* tokens in the review. We combine the attribute terms and the unstructured words in a review to create a bag-of-words for the classifier to use. We use linear-kernel SVMs (SVMLight[4]) with default parameters We denote this as **DC-Base**).

Entity Identification Baseline: We use as our entity identification baseline the approach outlined in Eq. (2). This is in line with state-of-the-art entity identification techniques

[3] `http://www.godbole.net/shantanu/work/kdd08rep.html`

[4] http://svmlight.joachims.org/

[5] when applied to single tables. Recall that this only makes use of the attribute terms in the document and the candidate matches for them. It does not make use of the list of unstructured words in the document. It does not involve any tunable parameters. We denote this is **EI-Base**.

Joint Model: This is our joint model described in this paper. We denote this as **JM**. One very desirable aspect of the model is that, like the baseline above, it does not involve any tunable parameters. We run at most 4 EM iterations, since loglikelihood mostly stabilizes by then.

For the entity identification task, in addition to accuracy, we also consider *average decision entropy* as a metric. For any document having k entity choices, we measure *decision entropy* for it as $-\sum_{i=1}^{k} P(e_i|d) \log P(e_i|d)$, where $P(e_i|d)$ is the predicted probability for entity e_i. For any entity identification approach, we can then take the average over *decision entropy* for all documents as a performance measure.

5.2 Experiments on the IMDB dataset

The IMDB dataset[5] consists of movies, actors, actresses, directors and writers lists, among other details. To create our unstructured document collection, we crawled the IMDB site and downloaded the first 10 reviews (user comments) for the top 50 movies for 25 IMDB genres. This led to 12,500 reviews in all. On the structured side, we created a single movie table containing movie details with 6 columns for $Actor1$, $Actor2$, $Actress1$, $Actress2$, $Director1$, $Director2$, $Writer1$ and $Writer2$. We use the actor rank information available in IMDB to populate the 2 actor (and actress) columns with the top 2 actors and actresses for the movie. No ranking being available for directors and writers, we pick the first two that are listed for the movie. Note that in order to make the entity identification task challenging, we *do not* include the movie name as a column. We populated this movie table with the top 1250 movies from the 25 genres. Additionally, we added 25,000 randomly drawn movies from IMDB spanning all 25 genres. This resulted in a total of 26,250 movies in our structured database. One issue with the IMDB dataset is that each movie is associated with multiple genres. We get around this issue by randomly picking one of the genres for each movie and repeating this over several runs for all experiments.

Pre-processing: Each review document was processed to obtain its list of unstructured words and the list of attribute terms and their corresponding column matches in the database. The list of unstructured words is found simply by removing stop-words from the document. To find the attribute terms from the document, we need to annotate mentions of actors, actresses, directors and writers from movie reviews. We use a state-of-the-art rule-based named entity annotator [18] to annotate all person names. For this, we added a set of regular-expression based rules that uses capitalization patterns combined with First and Last Name dictionaries. To ensure high recall, we configured the dictionaries by adding all the first and last names of actors, actresses, directors and writers in the IMDB database.

Next, we compute the candidate matches for each attribute term that we extracted by matching against the six '*person name*' columns in our database. As already mentioned, we need to account for name variations and spelling mistakes. Therefore, we need to perform similarity matching between attribute terms and database values. We experimented with various string similarity measures from the SecondString package[6]. In our final experiments, we used the *Soft-TFIDF* measure coupled with the Jaro-Winkler metric. We built a wrapper around it to handle initializations, etc. for names. To address similarity matching in a scalable fashion, we indexed database names using first and last name tokens. This helps in quickly looking up candidate matches from the database for each attribute term. We then compute the similarity scores for these pairs.

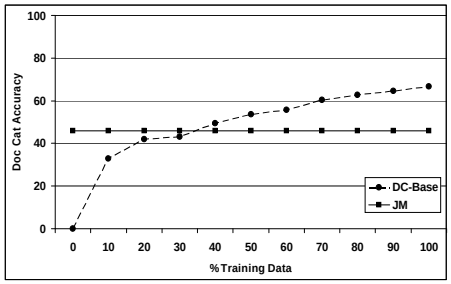


Figure 5: Doc categorization accuracy on IMDB

JM Acc	EI-Base Acc	**JM** Entropy	EI-Base Entropy
40.8%	38.5%	0.067	2.359

Table 1: Ent Identification Performance for IMDB

Results for IMDB: We plot the results of our experiments on the IMDB dataset in Figure 5 and Table 1. First, in Figure 5, we compare document categorization accuracy of **JM** against that achieved from the SVM baseline (DC-Base). Here we measure SVM accuracy on a separate held out set containing 20% of the reviews. On the X-axis, we vary the amount of training data provided for training. Of course, **JM** being unsupervised, it's accuracy remains constant. Observe that the supervised baseline catches up with **JM** only when it is provided with $\sim$ 35% of the document collection as training samples. For **JM**, this 'supervision' comes for free via automatic identification of entities for the documents. As an added benefit, there are gains on the entity identification side as well, as can be seen from Table 1. We can see that **JM** improves entity identification performance over the baseline from 38.5% to 40.8% in terms of accuracy. *But* this is not the complete picture. For documents where **JM** is not able to correctly nail down the right entity, it is nevertheless greatly successful in reducing the confusion over the possible choices. This can be seen by comparing the the average decision entropy over candidate entities, which is reduced to 0.067 for **JM** from 2.359 for the baseline. Much of this improvement is made possible by correctly inferring the type value (genre) for the movie entity, though the actual movie could not be identified. Note that this is in perfect agreement with the intuition that we formalized at the end of Section 3.

Though **JM** shows some improvement in entity identification for IMDB, it is greatly below our expectation. This is in part because enough matches could not be found for the

[5] `http://www.imdb.com/interfaces`

[6] `http://secondstring.sourceforge.net/`

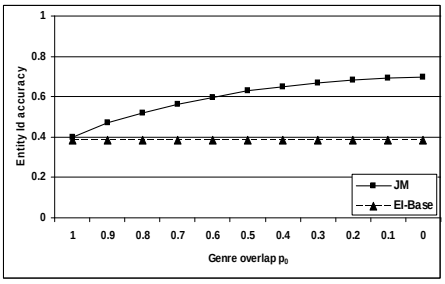
attribute terms in the documents. Recall that our back-end movie database does not have the names of the movies, and only contains names of the two actors and actresses, directors and writers. It is *quite challenging* to identify movies based on just this information, as the baseline performance shows. We expected **JM** to improve over this by correctly identifying the genres based on the words in the reviews. But the multiple genres originally associated with each movie turns out to be a big hindrance for this. Recall that for each run, we randomly picked one genre for the movie. (Numbers reported in Figure 5 and Table 1 are averaged over 10 runs) This results in significant overlap across genres in terms of words, and makes genre identification very challenging for **JM**. Note that DC-Base achieves only $\sim 65\%$ accuracy, even after using all the training data.

In summary, IMDB satisfies our motivational setting where words are related to column values in the database, but is not a perfect fit because of the multiple genre values associated with each movie. Unfortunately, the enterprise data that we worked on cannot be publicly disclosed, and we are not aware of publicly available data that has both the structured component and unstructured documents labeled with structured entities (that we need for evaluation). As a viable alternative, we experimented on semi-synthetic data, where we can control the degree of overlap between different categories and evaluate our model under different settings. This also provides insights into how our model is expected to perform in domains with varying characteristics. Next, we describe the semi-synthetic data generation process and our experiments on it.

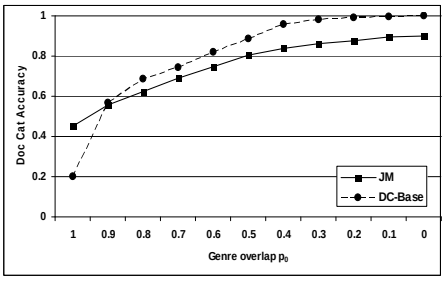
5.3 Experiments on Semi-Synthetic Data

In our semi-synthetic data, we keep the structured movie table untouched. Of the list of genres available for each movie, we randomly pick one in each run of the experiment, as before. On the unstructured side, we re-create the documents in two stages. First, we artificially partition the words in our vocabulary between the genres with different degrees of overlap. This is achieved using an overlap probability p_o. Each word in the vocabulary is first assigned to one randomly chosen genre from the list. Additionally, it is assigned to each of the remaining genres with probability p_o. Clearly, higher values of p_o result in the same word belonging to more genres, leading to less separability between genres. For each genre, we assume all words assigned to it to be equally likely. Once the words have been assigned to genres, we re-create the reviews in our collection. For each review, we keep the n attribute terms untouched. We replace the m original unstructured words with m words sampled uniformly from the genre of the corresponding movie. Effectively, only the unstructured words in each review are replaced by new words. Everything else stays exactly the same as before.

In our experiments on semi-synthetic data, we vary the degree of overlap between the words assigned to different genres, and analyze the performance of **JM**. We first consider the entity identification results plotted in Figure 6(a). We can see that for very high overlap among the words in different genres, improvement in entity identification performance over the baseline is minimal. This mirrors the original IMDB scenario, where the words do not have enough information for genres to be correctly assigned to reviews. However, as the words begin to get spread out more clearly between genres, entity identification performance improves significantly.



(a)



(b)

Figure 6: (a) Entity Identification and (b) Document Categorization results on semi-synthetic data

In fact, it increases by as much as 55.8% when words have medium overlap (from 38.5% to 60% for $p_o = 0.6$), and by 80.3% (from 38.5% to 69.4% for $p_o = 0.1$) when they are clearly indicative of the genre. This clearly underlines the huge potential that categorization using words lends to entity identification. In Figure 6(b), we compare the document categorization performance of **JM** against that of DC-Base. Baseline performance also naturally improves when words are better separated between genres, but the crux is that document categorization happens *for free* in **JM**, where as 80% training data is used for the baseline. Most interestingly, for large overlap between genres, **JM** actually *outperforms* the baseline in categorizing documents by leveraging attribute matches with the structured database. This is precisely the mutual reinforcement that we had expected from the model. We had explained this intuitively at the end of Section 3, and these empirical results lend further credence to our claims. In summary, these experiments clearly show that the two tasks can benefit each other immensely when conditions are favorable, which we believe happens naturally for many enterprise tasks.

6. RELATED WORK

One related body of work looks at integrating data across different structured data sources[17]. Several rule based and learning based techniques have been proposed for *schema matching*[14] that aims at matching related schema, and entity resolution/deduplication/tuple matching [9, 20, 1] that focuses on matching individual duplicate records in databases. Some effort has also gone into integrating information across unstructured resources[12, 13]. There has been some recent research on identifying structured entities from unstructured documents using pre-defined contexts[5]. However, the words in the documents that do not match against the database are ignored. In contrast, our goal is to use the

unstructured words in the document in addition to the attribute terms to identify structured entities.

One way to deal with the issue of labeled data is unsupervised clustering using generative models. Topic mixture models of increasing complexity have been proposed for documents over the years [15, 10, 2]. In comparison, we use a relatively simple model for the unstructured words in the document. It will be interesting to investigate if more complex mixture models can bring further improvements for entity identification. Also, all these generative models consider only unstructured words in documents, whereas we look to generate noisy entity mentions as well. The PRM framework [8] is a probabilistic generative model over values in a structured database that takes into account dependencies across different columns. Our focus is on documents that mention these values rather than the back-end data itself.

At a high level, the idea of using *information* or *knowledge* from one learning task to help another has been used in various forms. Semi-supervised methods are the most popular among approaches that bootstrap learning on insufficient labeled data by exploiting abundance of unlabeled data[16]. Co-training in multi-view learning [3] uses two independent feature partitions (views) on the same label-set and training data. The goal is to iteratively add training data to one classification task and re-learn models using the current labeling from the other. In co-clustering[6], two clustering tasks reinforce each other. More specifically, co-clustering looks to simultaneously cluster instance rows and feature columns by iteratively transferring knowledge from the rows to the columns (and vice-versa). Cross-training[19] exploits mappings between classes across related but non-identical classification tasks to overcome lack of training data in any individual task. Multi-task learning aims to improve generalization for a learning task by using higher order domain knowledge from related tasks. It has been used for clustering[21] where similarity measures are tweaked according to those in related clustering tasks, and in neural networks[4].

Our proposed model for joint entity identification and document categorization is similar in spirit, but unique in that we combine two hitherto unrelated tasks. We show how structured entity identification can benefit from unstructured words in documents via document categorization. Conversely, we also perform document categorization without an explicit label-set or training data where mapping documents to entities implicitly provides supervision via the column values in the database.

7. CONCLUSIONS

In summary, we have considered two critical tasks over document collections, namely entity identification and document categorization. We have observed that the two tasks can be related naturally in many domains so that they can mutually benefit each other. We have explored different ways to formally relate the two tasks and proposed, as a solution, a probabilistic model for documents that generates both entity mentions and unstructured words. We demonstrated that this model not only competes with supervised document classification approaches, but can also improve entity identification accuracy immensely in many scenarios. We believe that this paper will open up new directions of research in integration and analysis over unstructured document collections.

8. REFERENCES

[1] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM TKDD*, 1(1), March 2007.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

[3] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, 1998.

[4] R. Caruana. Multitask learning. *Machine Learning*, 28(1), 1997.

[5] V. T. Chakaravarthy, H. Gupta, P. Roy, and M. Mohania. Efficiently linking text documents with relevant structured information. In *VLDB*, 2006.

[6] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *SIGKDD*, 2001.

[7] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, 2005.

[8] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*, 3:679–707, 2003.

[9] M. Hernández and S. Stolfo. The merge/purge problem for large databases. In *SIGMOD*, 1995.

[10] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, 1999.

[11] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *ECML*, 1998.

[12] X. Li, P. Morie, and D. Roth. Semantic integration in text: from ambiguous names to identifiable entities. *AI Mag.*, 26(1), 2005.

[13] A. McCallum and B. Wellner. Conditional models of identity uncertainty with application to noun coreference. In *NIPS*, 2004.

[14] R. E. Melnik S., Garcia-Molina H. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, 2002.

[15] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2-3), 2000.

[16] K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3), 2000.

[17] B. P. A. Rahm Erhard. On matching schemas automatically. In *VLDB Journal*, volume 10(4), 2001.

[18] G. Ramakrishnan. *Bridging chasms in text mining through Word and Entity Associations*. PhD thesis, IIT Bombay, 2005.

[19] S. Sarawagi, S. Chakrabarti, and S. Godbole. Cross-training: Exploiting probabilistic mappings between topics. In *SIGKDD*, 2003.

[20] P. Singla and P. Domingos. Object identification with attribute-mediated dependences. In *PKDD*, 2005.

[21] S. Thrun and J. O'Sullivan. Discovering structure in multiple learning tasks: The tc algorithm. In *ICML*, 1996.