A Translation Model for Matching Reviews to Objects

Nilesh Dalvi Ravi Kumar Bo Pang Andrew Tomkins
Yahoo! Research
701 First Avenue
Sunnyvale, CA 94089.
{ndalvi,ravikumar,bopang,atomkins}@yahoo-inc.com

ABSTRACT

We develop a generic method for the *review matching* problem, which is to match unstructured text reviews to a list of objects, where each object has a set of attributes. To this end, we propose a translation model for generating reviews from a structured description of objects. We develop an EM-based method to estimate the model parameters and use this model to find, given a review, the object most likely to be the topic of the review. We conduct extensive experiments on two large-scale datasets: a collection of restaurant reviews from Yelp and a collection of movie reviews from IMDb. The experiments show that our translation model-based method is superior to traditional tf-idf based methods as well as a recent mixture model-based method for the review matching problem.

Categories and Subject Descriptors. I.2.7 [**Computing Methodologies**]:Natural Language Processing—*Language models*

General Terms. Algorithms, Experimentation, Measurements

Keywords. Language model; review matching; translation model

1. INTRODUCTION

Reviews are ubiquitous on the web — they exist in a variety of places on the web including online e-commerce websites (e.g., `amazon.com`), verticals (e.g., `imdb.com`), dedicated review websites (e.g., `yelp.com`), aggregation sites (e.g., `nextag.com`), blogs, forums, newspapers, and so on. With an increased pressure on search engines to present a holistic view of search results than mere ten blue links in response to a user query, it becomes critical for them to collect and understand user-generated content such as reviews. This will be especially useful for queries for which many web reviews exist — such queries include those related to shopping, dining, products, movies, etc. An important part of this understanding necessitates deciphering the object that is being reviewed, in other words, the search engine is faced with the *review matching* problem. Solving this problem is the stepping stone to enabling more sophisticated applications such as review aggregation, sentiment and polarity analysis, and more (Section 2.3).

At first glance, review matching seems a non-problem: after all, any reasonable review should have information about the object that is reviewed. Unfortunately, this turns out to be deceptive for two main reasons: review webpages might mention other objects that are peripheral to the review and the information revealed about the object in a review might be partial or surrogate; we discuss these issues at length in Section 2.1. In fact, these reasons make the review matching problem not amenable to vanilla techniques based on information retrieval, entity matching, or clustering; more on this in Section 2.2.

Our contributions. In this paper we explore the problem of matching reviews to objects using a translation model. Given a set of objects where each object has a set of attributes, we posit a simple model for generating a word in the review for the object from its attributes according to the following process. The process first chooses an attribute (independent of the object), and then selects a word in the chosen attribute of the object, and finally outputs a translation of this selected word according to a global, attribute-dependent translation model.

We then obtain a method to estimate the parameters of the model using a training dataset consisting of a set of aligned reviews, i.e., a set of pairs of reviews and their corresponding objects. The parameter estimation is based on the Expectation Maximization algorithm, where the maximization step is a non-linear maximization problem solved using gradient descent. We use these learned parameters in order to find, given a review, the object that is most likely to have generated the review.

Our generative model and the parameter estimation method represent a substantial generalization of the mixture model for review matching proposed in [10]. (See Section 3 for a discussion of the differences.)

We then apply our review matching method on a dataset of over 83,000 restaurant reviews from Yelp and on a dataset of over 36,000 movie reviews from IMDb. For the restaurant reviews, we obtain more than 50% improvement over the tf-idf method and more than 28% improvement over the mixture model [10]. For the movie reviews, we obtain around 10% improvement over tf-idf and 4.5% over the mixture model. The improvements are directly due to the power of the translation model and fully exploiting object attributes.

Our methodology is applicable beyond review matching; for example, it can be used to extract the primary politician discussed in a political news article. Going further, our method can be used in matching unstructured text to structured objects whenever it is plausible to assume that the text is produced from the objects using a model similar to ours.

Organization. In Section 2, we illustrate what makes the problem hard. Section 3 reviews related work on language modeling, information extraction, and opinion mining. Section 4 presents our generative model and an EM-based method to estimate the parameters of the model, with Section 5 discussing certain practical considerations. Section 6 contains a description of the data used in our experiments. Section 7 presents our experimental results on restaurant review (Yelp) and movie review (IMDb) datasets. Section 8 contains concluding remarks.

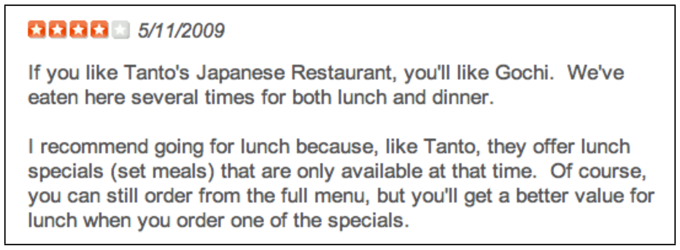
2. PROBLEM MOTIVATION

In this section we discuss three points that highlight the motivation for studying the review matching problem in depth. First, we ask what makes the problem difficult. Second, we address the issues with using existing technologies, from information retrieval to information extraction, to solve this problem. Third, we outline a set of applications that are enabled by solving the review matching problem.

2.1 Why is the problem hard?

It is most natural for a review to mention the object that it reviews. In fact, almost all review webpages typically contain a mention of the object that is reviewed in one form or another — either the object is mentioned in the review itself or is explicitly mentioned on the webpage that contains the review. Given this, how can the problem of review matching still be hard? We examine two main reasons:

(1) Review webpages may contain mentions of more than one object. This can occur in the review itself (e.g., the review compares the object to one or more other objects; see Figure 1). Alternatively, a webpage can contain reviews of more than one object (this is common in blogs and review aggregating websites, e.g., `http://tasteofthesuburbs.blogspot.com/search/label/chinese%2Fdim%20sum` contains a review of several chinese restaurants); or the reviews are about one single object, but the webpage can mention nearby or related objects as part of a website-wide template (this is common in review websites managed by content-generation software, e.g., `http://www.yelp.com/biz/gochi-japanese-fusion-tapas-cupertino` contains a list of nearby restaurants).

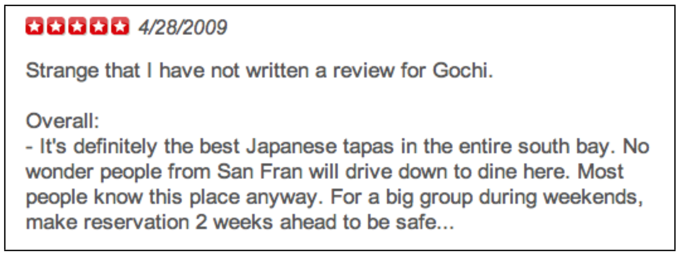

5/11/2009

If you like Tanto's Japanese Restaurant, you'll like Gochi. We've eaten here several times for both lunch and dinner.

I recommend going for lunch because, like Tanto, they offer lunch specials (set meals) that are only available at that time. Of course, you can still order from the full menu, but you'll get a better value for lunch when you order one of the specials.

Figure 1: A review snippet mentioning two restaurants.

(2) A review might mention an object in a partial manner or in a way that results in ambiguities. Reviews often tend to contain partial mentions (e.g., 'Gochi' instead of 'Gochi Japanese Fusion Tapas', the official name), abbreviate certain words (e.g., 'San Fran' instead of 'San Francisco' or 'Rd' instead of 'Road'), or use related terms (e.g., 'south bay' or 'bay area' instead of 'Cupertino'); see Figure 2.


4/28/2009

Strange that I have not written a review for Gochi.

Overall:
- It's definitely the best Japanese tapas in the entire south bay. No wonder people from San Fran will drive down to dine here. Most people know this place anyway. For a big group during weekends, make reservation 2 weeks ahead to be safe...

Figure 2: A review snippet with partial and substitute information.

Hence, any review matching method has to cope with both ambiguity and partial information that may be present on review webpages. One the other hand, a redeeming aspect of many review webpages is that they tend to contain clues about other attributes of the object. For example, a restaurant review may contain words that hint at the cuisine of the restaurant or its location. To achieve good performance, a review matching scheme should utilize such clues.

2.2 Problems with existing methods

Having examined the difficulties posed by the review matching problem, we consider the tempting proposition to repurpose existing methods to solve the review matching problem. Here we discuss the caveats.

Classical IR methods. It may appear that the review matching problem is an instance of the standard IR setting: the review is the query and the set of objects, along with their attributes, are the documents. Unlike in traditional IR, however, the query is long and the document is short; this stipulates adapting established IR concepts such as idf (inverse document frequency) and document-length normalization to this setting. A dual view of considering reviews as documents and objects as queries is still problematic since the goal is to find the best "query" for a given "document"; such a question is not explicitly addressed by classical IR. For a detailed discussion of these issues, see [10].

Wrapper induction and information extraction. If we manage to extract all the objects in the review through some means, then review matching might become simpler. There are two ways of extracting objects from webpages, namely, wrapper induction and information extraction. Wrapper induction constructs rules to navigate to certain portions of the HTML structure in order to extract objects (e.g., [16]). The main disadvantages of wrapper-based methods are twofold: they primarily apply to highly-structured websites and they involve considerable human labeling effort that is both expensive and prone to error. The latter is particularly undesirable since it is not a one-time cost: the wrappers have to be constantly kept up-to-date. The former places a limitation on the scope of the applicability of such methods; in particular, it is not feasible to study the less structured tail websites using wrappers. Information extraction methods, including named-entity recognition, often have limited accuracy [7, 23]. Even when they rely on an extensive dictionary lookup mechanism to identify object mentions in text, an additional co-reference resolution step might be necessary to resolve different mentions of the same physical object. In addition, when they extract multiple objects from a review, we are still left with the task of selecting one from among these candidates; this, for instance, will happen for the review snippet in Figure 1. In fact, the relatively large number of candidates yielded in the candidate generation phase in our experiments, which can be viewed as light-weight information extraction, suggest that this is a practical concern.

Classifier-based methods. Another possibility would be to try classifier-based methods to classify a review according to the values of various attributes. This calls for building a (multi-class) classifier for each of the attributes. Even if such classifiers are available, one needs to build additional layers on top of them to combine classifier output with other evidence present in the text, which is a non-trivial task in itself. In contrast, our model naturally combines all such evidence in a principled way.

2.3 Some applications of review matching

Matching reviews to objects is the first step in enabling many review-related applications. Below, we state three such applications.

(1) Fine-grained review localization. The problem here is to localize the review on a webpage such a blog, where the review is not the only piece of text present on the page. This can be done, especially in our language model-based method, by using the attributes of the object. Localized review text can help in many search engine tasks by indirectly removing noisy sections of a webpage. Likewise, our language model-based method can also be used to identify and tease out reviews of multiple objects in a single webpage.

(2) Review aggregation. The problem here is to aggregate the information contained in all the reviews for a given object. Aggregation might be as simple as computing the average rating of the object to as sophisticated as extracting the most hated aspect of the object by a detailed text analysis of the individual reviews.

(3) Automated information extraction. The task here is to extract structured information from webpages on a review website. By identifying the review objects on many webpages on a given website, it is possible to learn the HTML structure of the webpages (such as the HTML DOM node that contains various attributes of the object); this can be used in automatic information extraction.

3. OTHER RELATED WORK

The other related work falls into three main categories, namely, the topic of language modeling in general and its connections to information retrieval in particular, the literature on entity matching and information extraction, and the work on opinion topic identification.

Relationship to review language model [10]. In a recent work [10], we proposed a framework of using a review language model (RLM) to match reviews to objects. The specific instantiation explored in that work was a simple mixture model for generating reviews from a description of an object, which we refer to as MIXTURE in this paper. The current paper is a substantial generalization and extension of this earlier work. The salient differences between the two papers are the following.

(1) The model in this paper incorporates two generalizations over [10]: using the structured nature of the objects in an explicit fashion and using a translation model;

(2) Consequently, the parameter estimation method is significantly more complicated; for instance, there was no need for EM in [10];

(3) The experiments in this paper were conducted not only on a much larger restaurant review collection compared to [10] but also on movie review collection, obtained from IMDB, which is not present in [10].

The experiments in this paper firmly establish that our enhanced model indeed outperforms the basic MIXTURE model.

Language modeling. Language modeling has been a powerful paradigm in the context of several information retrieval applications [21, 24, 17, 27]. The principle behind this is to first estimate a language model for each document and for a given query, rank the documents in order of the likelihood of the query according to the estimated model of each document [21]. One of the main issues with language modeling is data sparsity; smoothing is an important means to manage data sparsity [9, 28]. Hofmann introduced probabilistic latent semantic indexing in which he proposed a mixture model with latent topics to generate words in documents [12]; while he also aims to go beyond exact word match, unlike in his case, our topics are patent: they are the attributes of the objects. Berger and Lafferty [3] introduced the idea of treating information retrieval as a statistical machine translation problem [5, 6]; our work is inspired by this translation viewpoint. Besides being in a non-IR setting and utilizing the structure in the objects, our setting also has access to more naturally aligned data, unlike in [3].

Information extraction and entity matching. Entity matching is a topic that is well studied in databases. There are three main approaches to entity matching, namely, non-relational, relational, and collective. Non-relational approaches consider pairwise attribute similarities between entities [19, 11]. Relational approaches exploit the relationships that exist between entities [1, 14]. And, collective approaches exploit the relationship between various matching decisions [4, 18]. The EROCS system [8], whose goal is to link structured data with unstructured text, is closest in spirit to our work. This system, based on information extraction and entity matching uses tf-idf for solving the matching problem. As our experiments establish, this is sub-optimal.

Opinion mining. Opinion topic identification is also somewhat related to our work. There has been a lot of research on opinion extraction from reviews [15, 26, 22, 13, 25]. These papers focus on finding the attributes of the object under review, rather than identifying the object itself. For the case when the objects are products, dictionary-lookup methods have limited success on general non-product review texts [25]. As we mentioned earlier, dictionary-lookup methods have limitations when applied to our problem: they can be more effective at identifying presence of object mentions than at disambiguating similar objects. There has been some work on finding reviews (regardless of their subjects) in large-scale collections [20, 2]; this is a logical step that precedes the review matching step.

4. MODEL AND METHOD

In this section we present the main technical content of the paper. First we start with the problem formulation and introduce the basic set up. We then present, in Section 4.2, a probabilistic model for generating reviews from objects. In Section 4.3 we briefly describe how to use this generative model for review matching and in Section 4.4 we describe a method based on Expectation Maximization (EM) to estimate the model parameters.

4.1 Problem formulation

Let $\mathcal{E}$ be a given set of objects. Each object $e \in \mathcal{E}$ has a set of attributes. For an object e, let e_k be the contents of its kth attribute; we treat the content of each attribute to be a bag of words. We use the notation $u \in e_k$ to denote that the word u is present in the kth attribute of e.

The *review matching* problem is the following: given a review which is mainly about one of the objects in $\mathcal{E}$, find the object that is being reviewed.

EXAMPLE 1. We use the following example from the restaurant domain as our running example. Let $\mathcal{E}$ be the set of restaurants. Each restaurant $e \in \mathcal{E}$ has three attributes associated with it: the `name` of the restaurant, the `city` where it is located, and the type of `cuisine` it serves. A particular instantiation of the restaurant object is given below:

`name`	Gochi Japanese Fusion Tapas Restaurant
`city`	Cupertino
`cuisine`	Japanese, Tapas

4.2 A generative model

We posit a simple model for generating a word in the review for an object from its attributes according to the following process. To begin, we describe it informally: the process first chooses an attribute (independent of the object), and then selects a word in the chosen attribute of the object, and finally outputs a translation of this selected word according to a global, attribute-dependent translation model.

Before formally defining it, we illustrate the model using the restaurant object from Example 1. In this example, a review about Gochi restaurant might be composed of words generated from its `name` (words such as *gochi*, *restaurant*) its `city` (*cupertino*), or its `cuisine` (words such as *japanese*, *tapas*). These words sometimes appear verbatim, and sometimes get *translated* into other words. For instance, *cupertino* (`city`) might generate words such as *south bay* or *bay area* and *japanese* (`cuisine`) might generate words such as *croquette* or *unagi*. In fact, any word in the review can be accounted for by this translation: for example, a generic word such as *is* or *of* can be translated from any word in any attribute of any object.

Now we describe the model formally. Let K denote the set of attributes of the objects and let U denote the set of all possible words in object attributes. Let V denote the vocabulary of reviews. Let α be a probability distribution over K. We use α_k to denote the probability of choosing attribute k. For each $k \in K$, let β_k be a function that assigns a positive real weight $\beta_k(u)$ to each word u in U. Finally, for each $k \in K$ and $u \in U$, let $t_k(\cdot \mid u)$ be a distribution over V such that the probability of a word w being translated from u is given by $t_k(w \mid u)$. The parameters α, β, and t are collectively referred to as θ.

Given parameters θ, a word w in review is generated from a word in the attribute content of an object e as follows. First, an attribute k is picked with probability α_k. Then, a word u is picked from the set e_k with probability proportional to $\beta_k(u)$; this is given by probability $\beta_k(u \mid e) = \beta_k(u)/B_k(e)$, where $B_k(e) = \sum_{u \in e_k} \beta_k(u)$ is the normalizing factor for e. Finally, a word w is picked with probability $t_k(w \mid u)$. Note that the word w is generated from the attribute–word pair (k, u).

Thus, the probability of a word w being generated from an object e is given by

$$\mathbf{P}_\theta(w \mid e) = \sum_k \alpha_k \sum_{u \in e_k} \beta_k(u|e) \cdot t_k(w|u). \tag{1}$$

And the probability of a review r being generated from an object e is given by

$$\mathbf{P}_\theta(r \mid e) = Z(r) \cdot \prod_{w \in r} \mathbf{P}_\theta(w \mid e), \tag{2}$$

where $Z(r)$ is a normalizing constant depending only on the length of r.

EXAMPLE 2. We continue with the setting in Example 1. Suppose we have the following parameters

$$\alpha_{\texttt{name}} = 0.2, \quad \alpha_{\texttt{city}} = 0.1, \quad \alpha_{\texttt{cuisine}} = 0.7.$$

Then, to generate a review word for the Gochi restaurant, the attribute `cuisine` will be chosen with probability 0.7. Suppose the $\beta_{\texttt{cuisine}}$ values are as follows:

$$\begin{aligned}\beta_{\texttt{cuisine}}(\textit{japanese}) &= 0.4,\\ \beta_{\texttt{cuisine}}(\textit{tapas}) &= 0.1,\\ \beta_{\texttt{cuisine}}(\textit{italian}) &= 0.3.\end{aligned}$$

Then, given that the attribute `cuisine` was chosen for Gochi, *japanese* will be picked from the set {*japanese*, *tapas* } with probability $0.4/(0.4 + 0.1) = 0.8$. Finally, we look at the $t_{\texttt{cuisine}}(\cdot|\textit{japanese})$ values to pick a review word. Suppose we have the following:

$$\begin{aligned}t_{\texttt{cuisine}}(\textit{japanese}|\textit{japanese}) &= 0.5,\\ t_{\texttt{cuisine}}(\textit{croquette}|\textit{japanese}) &= 0.1,\\ t_{\texttt{cuisine}}(\textit{unagi}|\textit{japanese}) &= 0.3.\end{aligned}$$

Then, given *japanese*, *unagi* will be picked with probability 0.3. The final probability of generating *unagi* from (`cuisine`, *japanese*) is $0.7 \times 0.8 \times 0.3 = 0.168$.

4.3 Review matching using generative model

Given the review language model θ, matching objects to reviews is straightforward. For a review r, we want to output the most likely object e^* given by

$$e^* = \arg\max_e \mathbf{P}(e \mid r) = \arg\max_e \frac{\mathbf{P}(e)}{\mathbf{P}(r)} \cdot \mathbf{P}(r \mid e).$$

In the absence of any information, we assume a uniform distribution for $\mathbf{P}(e)$. (Additional information about objects, such as their rating/popularity, can be used to model $\mathbf{P}(e)$ more accurately.) From this, we get

$$e^* = \arg\max_e \mathbf{P}(r \mid e) = \arg\max_e \prod_{w \in r} \mathbf{P}_\theta(w|e). \tag{3}$$

4.4 Parameter estimation

In this section we describe the methodology to estimate the parameters of our generative model. Our training data consists of a set of aligned reviews, i.e., a set of pairs of reviews and their corresponding objects. For the sake of presentation, we treat the training data as a sequence of pairs (w, e) where w is a word from a review and e is the object of corresponding review. We use $w^{(i)}$ to denote the word in the ith pair and $e^{(i)}$ to denote the object in the ith pair.

Clearly, if we knew which attribute–word pair (k, u) in $e^{(i)}$ generated the word $w^{(i)}$, for all i, then the parameter estimation would be easy. However, such alignment information at a word level is not available to us as part of the training data. We therefore introduce a hidden variable, $\mu^{(i)}_{u,k}$, which denotes the event that $w^{(i)}$ is generated from the attribute–word pair (k, u). Consider the following function:

$$F(w, u, k; \theta, e) = \alpha_k \cdot \beta_k(u|e) \cdot t_k(w|u). \quad (4)$$

Then, we have

$$\mathbf{P}_\theta(w^{(i)}, \mu^{(i)}_{u,k}|e^{(i)}) = F(w^{(i)}, u, k; \theta, e^{(i)}).$$

Clearly,

$$\mathbf{P}_\theta(w^{(i)}|e^{(i)}) = \sum_{k,u} \mathbf{P}_\theta(w^{(i)}, \mu^{(i)}_{u,k}|e^{(i)}).$$

Our goal is to estimate θ so as to maximize

$$\prod_i \mathbf{P}_\theta(w^{(i)}|e^{(i)}),$$

or equivalently, to maximize

$$\sum_i \log \mathbf{P}_\theta(w^{(i)}|e^{(i)}).$$

We use the Expectation Maximization (EM) method to solve this maximization problem.

E-step. In the E-step, given the current model $\theta^{(t)}$, we compute

$$\mathbf{E}^{(t)}(\mu^{(i)}_{u,k}) = \frac{F(w^{(i)}, u, k; \theta^{(t)}, e^{(i)})}{\sum_{u',k'} F(w^{(i)}, u', k'; \theta^{(t)}, e^{(i)})},$$

using (4).

M-step. Since the original objective function is difficult to directly optimize for, we instead compute

$$\theta^{(t+1)} = \arg\max_\theta Q(\theta, \theta^{(t)}),$$

where

$$\begin{aligned} &Q(\theta, \theta^{(t)}) \\ &= \sum_i \sum_{k,u} \mathbf{E}^{(t)}(\mu^{(i)}_{u,k}) \log p_\theta(w^{(i)}, \mu^{(i)}_{u,k}|e^{(i)}) \\ &= \sum_i \sum_{k,u} \mathbf{E}^{(t)}(\mu^{(i)}_{u,k}) \log \left(\alpha_k \frac{\beta_k(u)}{B_k(e^{(i)})} t_k(w^{(i)}|u) \right). \end{aligned}$$

Using the Lagrange multipliers method, we compute the optimal parameters $\alpha^*_k, t^*_k, \beta^*_k$ of the new model $\theta^{(t+1)}$.

First,

$$\alpha^*_k = \frac{g_k}{\sum_{k'} g_{k'}},$$

where

$$g_k = \sum_{i,u} \mathbf{E}^{(t)}(\mu^{(i)}_{u,k}).$$

Next,

$$t^*_k(w|u) = \frac{f_k(u, w)}{\sum_{w'} f_k(u, w')},$$

where

$$f_k(u, w) = \sum_{i:w^{(i)}=w} \mathbf{E}^{(t)}(\mu^{(i)}_{u,k}).$$

Since we cannot obtain a closed form for β^*_k, we use the gradient descent method to estimate β^*_k. We obtain $\Delta\beta_k(u) =$

$$\begin{aligned} &\frac{\partial Q(\theta, \theta^{(t)})}{\partial \beta_k(u)} \\ &= \sum_i \frac{\mathbf{E}^{(t)}(\mu^{(i)}_{u,k})}{\beta_k(u)} - \sum_{i|u \in e^{(i)}_k} \frac{1}{B_k(e^{(i)})} \left(\sum_{u'} \mathbf{E}^{(t)}(\mu^{(i)}_{u',k}) \right). \end{aligned}$$

We solve the above non-linear equation using standard gradient descent. In fact, we can show that the solution is optimal; we omit the details in this version.

5. PRACTICAL CONSIDERATIONS

5.1 Regularization

Our set of equations is under-constrained, i.e., two different set of parameters can give rise to equivalent models. For instance, in the case of restaurants in Example 1, if there is a generic word w that is not correlated with any specific `city` or `cuisine`, we can distribute its probability among `city` and `cuisine` in any way we want and get the same model.

We say that $\theta \equiv \theta'$ if for all reviews r and objects e, $\mathbf{P}_\theta(r \mid e) = \mathbf{P}'_\theta(r \mid e)$.

THEOREM 5.1. *Let θ be a model such that for some k and w_0, $\min_u t_k(w_0 \mid u) > 0$. Then, there is a model θ' such that $\theta \equiv \theta'$ and $\min_u t_k(w_0 \mid u) = 0$.*

PROOF. Let $c = \min_u t_k(w_0 \mid u)$; note that $c \in (0, 1)$. Consider any attribute $\ell \neq k$. We will move the probability mass of w_0 from k to ℓ without affecting the model.

Define θ' as follows. Let $\Delta = c \cdot \alpha_k/\alpha_\ell$, and consider the following set of new parameters:

$$\begin{aligned} t'_k(w_0 \mid u) &= (t_k(w_0 \mid u) - c)/(1 - c) \\ t'_k(w \mid u) &= t_k(w \mid u)/(1 - c) \quad \text{for } w \neq w_0 \\ \alpha'_k &= (1 - c)\alpha_k \\ t'_\ell(w_0 \mid u) &= (t_\ell(w_0 \mid u) + \Delta)/(1 + \Delta) \\ t'_\ell(w \mid u) &= (t_\ell(w \mid u))/(1 + \Delta) \quad \text{for } w \neq w_0 \\ \alpha'_\ell &= \alpha_\ell + c\alpha_k \end{aligned}$$

All other parameters of θ' are same as those of θ. Clearly, $\min_u t'_k(w_0 \mid u) = 0$. One can verify that for any w and e, $\mathbf{P}_\theta(w \mid e) = \mathbf{P}'_\theta(w \mid e)$, i.e., $\theta \equiv \theta'$. We omit the details in this version. □

Theorem 5.1 shows that our system is under-constrained. It also suggests an approach to *regularize* the model as follows. We add an extra attribute `generic` to each object, which has a value *Review*. The intuition is that this attribute accounts for the generic review words that are not correlated with any specific attribute. For instance, in restaurant reviews, words like *tasty* and *dinner* do not have strong correlations with any specific city or cuisine. We want the attribute `generic` to account for all such words. Specifically, we want $t_{\texttt{generic}}(w \mid \textit{Review})$ to denote the probability that a word w is chosen from a generic review language.

We can use Theorem 5.1 to achieve this. For each attribute k and each word w such that $t_k(w \mid u) > 0$ for all u, we can take the minimum $t_k(u, w)$, subtract it from everyone and move it to the `generic` attribute.

Since regularization does not change the model, it does not affect the accuracy of the model in the matching task. However, it serves two purposes. First, it can make individual translation models, i.e., t_k cleaner since now they do not have to include generic words not related to the attribute. Second, the regularization can be used as a tool to avoid overfitting of the model, as we explain in the next section.

5.2 Avoiding overfitting

Recall that we introduce the `generic` attribute to account for generic review words. When we learn a model with this additional attribute, we find that $\alpha_{\texttt{generic}}$ gets driven down to zero after EM, i.e., the `generic` attribute gets no probability mass and the model prefers to explain even the seemingly generic terms using other object-dependent attributes. This is the result of overfitting. Consider a generic word like *dinner*. While all cuisines and cities should see the word with roughly the same frequency, it will not occur with the exact same frequency due to sampling errors. Thus, the model will let *dinner* get completely accounted for by words in `city` and `cuisine`, so as to account for the small differences in observed frequencies with different objects. Overfitting not only drives $\alpha_{\texttt{generic}}$ to zero, it also makes individual translation models noisy as sampling errors get amplified.

We use a very simple yet effective heuristic to overcome the problem of overfitting. While learning the model, we constrain $\alpha_{\texttt{generic}}$ to have a high value. Given this constraint, the `generic` attribute accumulates all the generic review words, and the individual translation models for specific attributes only account for words specific to them.

5.3 Using additional properties of attributes

We can improve model accuracy and efficiency by using additional knowledge about attributes, namely the scope of their translations. Certain attributes like `cuisine` and `city` can get translated to several words specific to the attribute, while certain other attributes like `name` or `phone number` can only translate to themselves. In general, if we know the set of words each attribute can translate into, we can incorporate the knowledge to learn a more effective model.

We use a very simple version of this idea. For each attribute, we label it as either *flexible* or *inflexible*. A word in a *flexible* attribute can be translated into any word in the vocabulary while a word in an inflexible attribute can only get translated into itself. By declaring attributes like `name` and `phone number` as inflexible, we can avoid learning complete translation models for all possible terms in such attributes, making models more compact and less noisy.

6. DATA

We consider two datasets, YELP and IMDB, from two different domains where objects have different types of attributes.

We obtained the YELP dataset based on the dataset used in [10]. The dataset consists of a set of reviews extracted from the Yelp website, `yelp.com` and the database of restaurant listings in Yahoo! Local, `local.yahoo.com`. The matching task is to determine, for each Yelp review, the corresponding restaurant listing in Yahoo! Local, where each restaurant has a `name`, `city`, and `cuisine` attribute. The dataset also contains the ground truth, i.e., the true mapping between Yelp and Yahoo! Local, which we use for training as well as for evaluation.

Some of the Yelp reviews do not contain any identifying information and can be as short as "Great place. Awesome food!!". In [10], only a subset of reviews, which explicitly contained the names and cities of restaurants that a human can use to uniquely identify the restaurant, were selected. However, a human can match a review even if it does not contain explicitly information, e.g., a mention of Manhattan can be used to infer that the restaurant is very likely in New York. In our evaluation, we consider all reviews, with the only requirement that they at least mention the restaurant name (even if only partially). As a result, our dataset is a superset of the one used in [10], and raw numbers are not directly comparable.

The resulting dataset consists of 83,478 Yelp reviews covering 8,006 unique restaurants, which we seek to match with 680,000 Yahoo! Local restaurant listings. We split the restaurants into training and test sets; 12,500 reviews on the training restaurants were set aside as training data, and 48,623 reviews are left in the test data.

The IMDB dataset consists of movie reviews from the IMDb website, `www.imdb.com`. For each movie, IMDb has a webpage that contains all the information about it. Also, each movie page has "User Comments" section where users submit reviews for the movie. We used hand-crafted extraction rules to extract all the movie information as well as reviews from IMDb. We extracted 36,321 reviews covering 3,786 randomly selected movies.

Our task is to match these reviews against the complete IMDb database of 156,355 movies. For each movie, we use `name`, `year`, `genre`, `director`, `actor list`, and `cast list` as attributes. Since the reviews were extracted from the IMDb for the corresponding movies, we know the true match for each review. We set aside half of the reviews for training and use the other half for evaluation. The split was done in such a way that movies do not overlap across the two sets.

Note that unlike the restaurant domain, a movie can often be uniquely identified by a combination of different attributes even when its `name` is not explicitly mentioned in the review.

7. EXPERIMENTS

We first describe the performance of our method on the two datasets in Section 7.1. Then, in Sections 7.2 and 7.3, we discuss in depth the models we learn on the YELP and IMDB datasets respectively, and analyze how different aspects of our modeling help improve the accuracy of review matching.

7.1 Performance on the review matching task

We refer to our review matching method using translation models as TMODEL. We compare it with two other methods: (i) the TFIDF method, which uses the classic tf-idf score between the review and the objects and (ii) the MIXTURE method [10], which uses a simpler instantiation of a mixture model to match reviews to objects and has been shown to outperform TFIDF in certain cases.

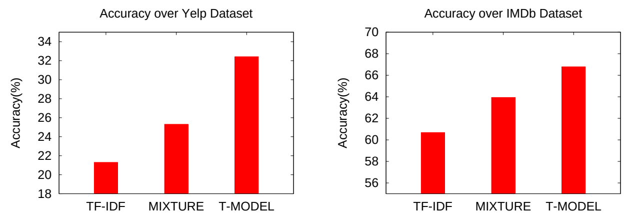


Figure 3: Accuracy of TFIDF, MIXTURE, and TMODEL on YELP and IMDB data.

Figure 3 shows the accuracy of top-one predictions from all three methods on YELP and IMDB. As we see, TMODEL obtains more than 28% improvement in accuracy over MIXTURE and more than 50% improvement over TFIDF. Note that the absolute accuracy numbers are low in general for the YELP dataset. This is because a large fraction of reviews do not have identifying information, and the maximum accuracy that can be achieved even by a human will be substantially lower than 100%. Likewise, we see that TMODEL improves over both MIXTURE (4.5%) and TFIDF (10%).

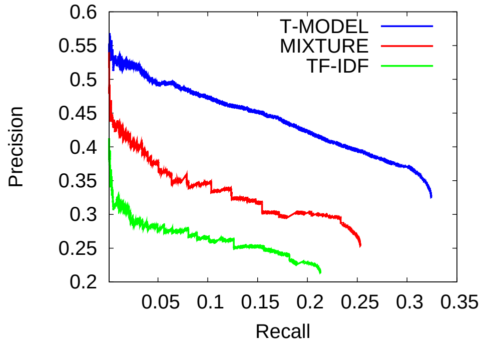
We then look at all the (review, top-one match) pairs, sorted by the score given by (3). Note that the reviews can be of different lengths and hence we normalize the scores by scaling them by $\prod_{w \in r} \mathbf{P}_\theta(w|\texttt{generic})$. Figures 4(a) and 4(b) present the precision–recall curves. We observe the same overall trends as in the case of accuracy: TMODEL outperforms MIXTURE, which in turn outperforms TFIDF.

7.2 Food for thought

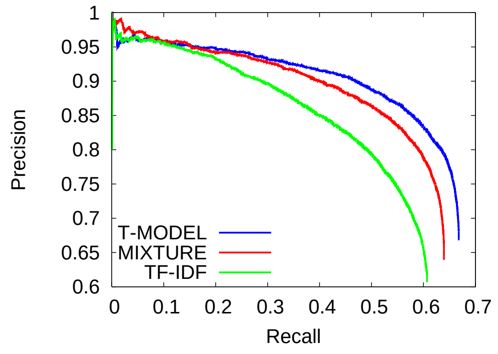
In this section we discuss the model learned on the YELP data in more detail. We start by discussing the models learned for the two flexible-match attributes: `cuisine` and `city`.

As we will see from the example translation tables, while initially both attributes receive all of the words in a review as candidate translations, in the final model, top translations for words in the `cuisine` attribute are predominantly food-related, while top translations for words in the `city` attribute are predominantly location-related. In other words, as a natural outcome of optimizing for the maximum likelihood of the data, food-related (and object-dependent) words are mostly accounted for by the `cuisine` attribute, and location-related words are mostly accounted for by the `city` attribute.

On `cuisine`. Recall that according to our model, words with higher β_k values are more likely to be chosen from attribute k. Intuitively, it is advantageous for more salient words — tokens that better distinguish an object from other objects — to receive higher β values, so that more generic words in reviews are left to be explained by the generic review language, and the salient features of the object can "concentrate" on explaining more object-dependent words.



(a) YELP (48,623 reviews)



(b) IMDB (17,878 reviews)

Figure 4: Precision–recall of TMODEL, MIXTURE, and TFIDF on the test data in YELP and IMDB.

We now examine the cuisine type attribute of restaurant objects to see whether the model learned conforms to our intuition.

In the YELP aligned data (Yelp reviews aligned with Yahoo! Local listings), there are 96 cuisine types with at least five restaurants represented in the data. A generic `cuisine` type *restaurants* is associated with all of the objects. Table 1 presents the most and least frequent cuisines types, as well as cuisine types with the highest and lowest β values. Figure 5 plots the β value assigned to different cuisine types against the frequency of that cuisine type in the data (i.e., number of restaurants associated with that type).

Intuitively, cuisine types that are associated with many reviews tend to be more generic types that are not very salient. Indeed, $\beta_{\texttt{cuisine}}(\textit{restaurants})$ is an order of magnitude lower than all the other β values. Thus, when an object is not associated with any specific cuisine type, when words are generated from this attribute, they will be drawn entirely from the *restaurants* translation table. On the other hand, when an object is associated also with a specific cuisine type, words for this attribute will be predominantly drawn from the more specific one, given the striking difference in the β values.

This trend also holds true for the non-generic cuisine types. For instance, the more general cuisine type *southeast_asian* is seen more frequently than the more specific cuisine type *vietnamese*. As we can see from Table 1, $\beta_{\texttt{cuisine}}(\textit{southeast\_asian})$ is among the lowest, while $\beta_{\texttt{cuisine}}(\textit{vietnamese})$ is among the highest. Thus, if an object is labeled as both *southeast_asian* and *vietnamese*, given that $\beta_{\texttt{cuisine}}(\textit{vietnamese})$ is high, it is

most frequent	*restaurants, american, clubs, bars, italian, seafood, continental, bars_and_pubs, french*
least frequent	*buffets, casinos, crepe, ethiopian, food_delivery, marketing_agencies, natural_and_organic_foods, swiss, dim_sum*
highest $\beta_{\texttt{cuisine}}$	*indian, german, pizza, malaysian, thai, barbecue, japanese, greek, vietnamese, cuban*
lowest $\beta_{\texttt{cuisine}}$	*restaurants, american, continental, southeast_asian, pubs, clubs, bars, catering_services, seafood*

Table 1: Example cuisine types

indian (70)	german (20)	japanese (160)	greek (36)	cuban (21)	moroccan (8)	healthy (55)	italian (333)	french (198)	american (717)
indian	german	sushi	greek	cuban	moroccan	organic	italian	french	shabu
naan	schnitzel	japanese	gyro	mojitos	couscous	bookstore	pasta	france	ye
masala	sauerkraut	roll	pita	vieja	tagine	macrobiotic	italy	croque	dog
tikka	germany	tempura	greece	ropa	bastilla	healthy	gnocchi	frites	sliders
tandoori	bratwurst	sashimi	gyros	leche	dancer	vegan	linguine	foie	pastrami
paneer	wiener	miso	feta	cubano	powdered	cobb	rigatoni	sous	stripper
buffet	sauerbraten	nigiri	tzatziki	cane	morocco	branzino	alla	souffle	grits
samosas	jaegerschnitzel	rolls	moussaka	cuba	casablanca	salads	pastas	provence	etouffee
india	spatzle	maki	opa	arroz	baklava	australia	spaghetti	sweetbreads	ogden
saag	spaetzle	sake	saganaki	dulce	tagines	flatiron	di	gras	officer
korma	suppenkuche	japan	taverna	guava	marrakech	grass	chianti	pommes	rink
palak	hefeweizen	udon	flaming	mojito	marrakesh	emerald	bruschetta	rillettes	creams
vindaloo	germans	eel	greeks	versailles	carrots	grains	antipasti	tarte	lanes
lassi	spatzel	izakaya	baklava	havana	hips	quinoa	antipasto	pate	bradley
chaat	bier	kaiseki	phoenicia	miami	clark	noon	mozzarella	christophe	duties
biryani	liter	hamachi	souvlaki	ajiaco	filo	movement	bolognese	michelin	4-course
rogan	wursts	seaweed	dolmas	leches	baba	wonton	tiramisu	paris	winemaker
malai	berlin	soba	horiatiki	gracias	munch	eleven	parmigiana	je	dogs
aloo	liquors	agedashi	radishes	latin	tabbouleh	habits	secondi	parisian	trout
samosa	munich	unagi	spanakopita	cha-ching	dimly	cliche	peroni	'aime	reuben

Table 2: Examples from translation tables for the `cuisine` attribute: top words w that are translated from u = different cuisine types, shown from left to right in decreasing order of $\beta_{\texttt{cuisine}}(u)$ (frequency of `cuisine` in the aligned data shown in parenthesis).

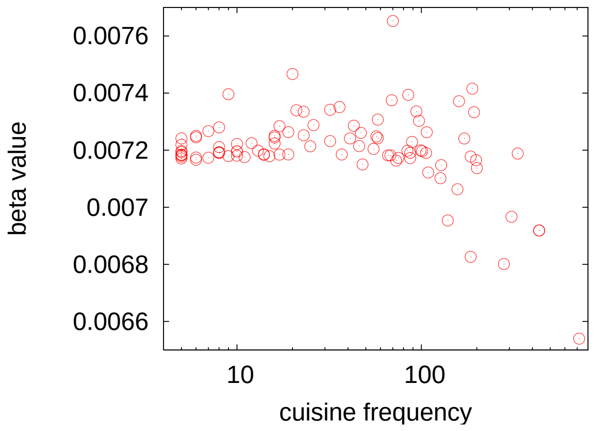


Figure 5: $\beta_{\texttt{cuisine}}$ (saliency of a cuisine type in the model) vs. cuisine frequency (number of restaurants associated with a cuisine type in the aligned data) for all cuisine types with frequency ≥ 5 (the most generic cuisine type *restaurants* is omitted.)

more likely to be chosen to generate words for reviews written about this object. Indeed, many of the most frequent cuisine types (see line 1 of Table 1) were assigned the lowest β values in the model (see line 4 of Table 1).

Very infrequent cuisine types are not necessarily all salient (see line 2 of Table 1 for examples). However, they may appear to be salient as a result of overfitting to the few reviews associated with them. On the other hand, the salient ones among them do not receive enough training data to reach really high β values. In effect, infrequent cuisines were mostly assigned mid-range β values (Figure 5).

Most of the most salient cuisine types turned out to be "mid-size" cuisines: cuisines not too general that they are used to describe too many restaurants, but still popular enough to receive enough training data. Thus, the area surrounding the data-points in Figure 5 roughly assumes a triangle (or sickle, to be more accurate) shape.

Furthermore, we can examine the translation table for each cuisine type. Table 2 presents tokens with the highest t values for selected cuisine types, in decreasing order of their β values.

First, perhaps not surprisingly, the most likely token for most cuisine types is the country of origin. One exception is *Japanese*, where, at least in the YELP data, the word *sushi* is apparently more representative of *Japanese restaurants* than the word *Japanese* (which comes in as second). On the other hand, the *American restaurants* as a label used in Yahoo! Local data seems to lack a clear definition: the word *American* is not even among the top translations. Its lack of identity is clearly reflected in both the low $\beta_{\texttt{cuisine}}$ value and the lack of focus in its translation table. *Healthy* is a quasi-cuisine-type that did not have an existence as distinctive as other cuisine types listed here. While its translation table is not quite as focused, we still observe reasonable translations such as *organic, vegan, grains* that are indeed quite representative of this type of food. In general, we observe both distinctive food items (*naan, masala* for *Indian*) and cuisine-related geographic terms (*havana, miami* for *Cuban*) in these top translations.

Reasonable translations were learned for very infrequent cuisine types as well. For instance, only eight restaurants were associated with *Moroccan* in the aligned data, yet it received words with distinctively Moroccan taste (e.g., *couscous, tagine*) as its top translations. Note that even those cuisines with relatively lower β values (apart from *American*) received reasonable translations (e.g., *foie gras, souffle* for *French*).

On `city`. We now proceed to examine the translation tables learned for the `city` attribute.[1]

chicago	york	francisco	boston
chicago	york	francisco	boston
hashbrowns	nyc	sf	newbury
loop	midtown	san	pool
michigan	manhattan	mission	sox
lincoln	yorkers	bernal	ligaya
byob	meatpacking	belden	copley
dog	soho	castro	hanover
commander	jean-georges	wharf	fenway
frontera	chika	cha	hall
herring	uws	fisherman	profiterole
polish	ny	soju	regina
cubs	atif	soma	faneuil
lux	yorker	dat	boylston
halsted	ues	richmond	kenmore
devon	chelsea	geary	bostonians
chicago-style	fondue	danko	aujourd
navy	bukhara	embarcadero	beacon
swedish	fours	fillmore	chinatown
clark	posto	gd	maki
chicagoans	queens	soap	end

Table 3: Examples from translation tables for the `city` attribute: top words w that are translated from $u \in \{$*chicago, (new) york, (san) francisco, boston*$\}$.

chicago	boston	bay	seattle
chicago	boston	bay	seattle
park	newton	sausalito	bellevue
milwaukee	brighton	emeryville	redmond
madison	brookline	oakland	kirkland
evanston	allston	monterey	portland

Table 4: Examples from translation tables for the `city` attribute: top words u that translate into $w \in \{$*chicago, boston, bay, seattle*$\}$.

Table 3 shows examples of top words w that are translated from $u \in \{$*chicago, (new) york, (san) francisco, boston*$\}$. First note that for *(new) york* and *(san) francisco*, popular abbreviations (*nyc* and *sf*) were indeed found among top translations. In addition, when the physical location of a restaurant is in a metropolitan city, top translations often include neighborhood names (e.g., *manhattan, meatpacking, chelsea, queens* for *(new) york*), notable streets (e.g., *geary* for *(san) francisco* and arguably *michigan (ave)* for chicago), and tourist attractions (e.g., *fisherman, wharf* for *(san) francisco* and *copley, faneuil* for *boston*).

On the other hand, when the physical address of a restaurant is technically in a satellite city in a metropolitan area, names of the metropolitan cities often appear in the corresponding reviews and are discovered to be likely translations. Table 4 presents notable examples. For instance, words that are most likely to translate into *boston* (i.e., $u \in \{$*boston, newton, brighton, brookline, allston*$\}$) are all neighborhoods in the "Greater Boston" metropolitan area.

[1]Since the `city` attribute will have only one city in its value, and we normalize β value for each attribute separately, there is essentially no competition, and $\beta_{\texttt{city}}$ is uniformly distributed, thus not as interesting to examine.

On `name`. As we mentioned earlier, `name` is an inflexible-match attribute. Thus, there is no translation table learned: words picked (according to $\beta_{\texttt{name}}(u|e)$) will translate only to itself. However, the $\beta_{\texttt{name}}$ values are still worth a brief discussion. The main intuition that we hope to be captured by $\beta_{\texttt{name}}$ values is: certain common words such as 'restaurant' or 'cafe' are likely to be dropped when people refer to restaurant names in informal reviews. Indeed, our model captures this intuition. Words with the lowest $\beta_{\texttt{name}}$ values (in increasing order) are: *restaurant, incorporated, bar, cuisine, drive-in, ristorante, lounge, grill*, etc. Note at matching time, this leads to higher normalized $\beta_{\texttt{name}}(u|e)$ values to the other non-generic terms in `name`, thereby giving them higher weights in the `name` part of the matching score.

This is a good time to step back for a moment and reflect. One might think the above effect is trivial to achieve by simply reducing the weights (in the matching score) for common words in the corpus, which can be achieved by TfIdf. Unfortunately, this does not work. First, the β values do not monotonically decrease with the word frequencies: some relatively frequent words (such as 'thai') still receive high $\beta_{\texttt{name}}$ scores, which suggests that the model finds it more "profitable" to use the `name` attribute to account for such words. On the other hand, for words like 'restaurant', even when they are officially part of the name of the object, our model finds it more "profitable" to use the generic review language to account for them, leaving the mass of the probability from the `name` attribute to concentrate on other words that are best explained by this attribute. Through correct "attributions", such words are less likely to be picked from the `name` attribute in the end. Second, while lower β values are desired for some of the words with high collection frequency, this is not universally true and depends on different semantics behind different attributes. As we will see in Section 7.3, for the `actor` attribute of movie objects, the intuition is the exact opposite: words with higher frequency in collection (names of more popular actors) are more likely to be picked than words with lower frequency in collection (names of unknown actors).

7.3 Notes on movies

Our model provides a fairly general framework. Even though movie objects are very different from restaurant objects, the underlying inference process remains the same.

In this section we briefly examine two attributes of movie objects. The first is a flexible-match attribute `genre`, where the results largely corroborate our findings on the Yelp dataset in that reasonable translation tables are learned. The second is an inflexible-match attribute `actor`, where the β values learned provide interesting contrast to our findings on the `name` attribute of restaurant objects.

On `genre`. Table 5 presents examples from translation tables for the `genre` attribute. Top translations for selected genres are summarized.

On `actor`. As briefly mentioned in discussions on `name` in Section 7.2, higher $\beta_{\texttt{actor}}$ values are observed for well-known

animation	western	musical	horror	crime	romance	thriller	comedy
animation	westerns	musicals	horror	crime	romantic	thriller	comedy
animated	western	busby	gore	gangster	her	action	funny
disney	stagecoach	musical	slasher	noir	she	horror	hilarious
pixar	cattle	rhps	zombie	cop	love	suspense	jokes
cartoon	gunfighter	dubin	zombies	heist	romance	plot	comedies
animators	leone	broadway	vampire	mob	comedy	killer	laugh
cartoons	apaches	berkeley	vampires	police	woman	thrillers	humor
dreamworks	saloon	bjork	scary	detective	chemistry	effects	laughs
looney	outlaw	numbers	creepy	mafia	girl	tension	funniest
voiced	derringer	songs	scares	murder	relationship	bad	her

Table 5: Examples from translation tables for the genre attribute: top words w that are translated from u = different genres.

actors (e.g., *bogart, hepburn, nicholson, pacino, hanks*), who appear more frequently in reviews, and lower $\beta_{\texttt{actor}}$ values are observed for unknown actors, who appear less frequently in reviews. This conforms to our intuition of the likelihood of actors being picked from cast members when a movie is being discussed in a review, but is in stark contrast to the case of restaurant `name` attribute, where the more frequent words were assigned lower $\beta_{\texttt{name}}$ values. As we noted earlier, even though the underlying inference process is the same, the different semantics behind different attributes lead to differences in the parameters quite naturally.

8. CONCLUSIONS

We developed a generic method for the review matching problem. We proposed a statistical translation model that incorporates the structured description of objects, for generating reviews. The parameters of the model were estimated using an EM algorithm. This model was used to find, given a review, the object most likely to be the topic of the review. We conducted experiments on two real-world datasets, namely, a restaurant review collection from Yelp and a movie review collection from IMDb. Our experiments showed that the translation model is superior not only to traditional tf-idf based methods but also to a recent mixture model-based method for the review matching problem.

9. REFERENCES

[1] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *Proc. 28th VLDB*, pages 586–596, 2002.

[2] L. Barbosa, R. Kumar, B. Pang, and A. Tomkins. For a few dollars less: Identifying review pages sans human labels. In *Proc. NAACL*, pages 494–502, 2009.

[3] A. Berger and J. Lafferty. Information retrieval as statistical translation. In *Proc. 22nd SIGIR*, pages 222–229, 1999.

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

[5] P. Brown, J. Cocke, S. D. Pietra, V. D. Pietra, F. Jelinek, J. Lafferty, R. Mercer, and P. Roosin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

[6] P. Brown, S. D. Pietra, V. D. Pietra, and R. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

[7] C. Cardie. Empirical methods in information extraction. *AI Magazine*, 18(4):65–80, 1997.

[8] V. T. Chakaravarthy, H. Gupta, P. Roy, and M. Mohania. Efficiently linking text documents with relevant structured information. In *Proc. 32nd VLDB*, pages 667–678, 2006.

[9] S. Chen and J. Goodman. An empirical study of smoothing technique for language modeling. Technical Report TR-10-98, Harvard University, 1998.

[10] N. Dalvi, R. Kumar, B. Pang, and A. Tomkins. Matching reviews to objects using a language model. In *Proc. EMNLP*, pages 609–618, 2009.

[11] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *JASIS*, 64:1183–1210, 1969.

[12] T. Hofmann. Probabilistic latent semantic indexing. In *Proc. 22nd SIGIR*, pages 50–57, 1999.

[13] M. Hu and B. Liu. Mining opinion features in customer reviews. In *Proc. AAAI*, pages 755–760, 2004.

[14] D. V. Kalashnikov, S. Mehrotra, and Z. Chen. Exploiting relationships for domain-independent data cleaning. In *Proc. 5th SDM*, 2005.

[15] N. Kobayashi, K. Inui, Y. Matsumoto, K. Tateishi, and T. Fukushima. Collecting evaluative expressions for opinion extraction. In *Proc. 1st IJCNLP*, pages 596–605, 2004.

[16] N. Kushmerick, D. S. Weld, and R. B. Doorenbos. Wrapper induction for information extraction. In *Proc. 15th IJCAI*, pages 729–737, 1997.

[17] J. Lafferty and C. Zhai. Probabilistic relevance models based on document and query generation. In W. B. Croft and J. Lafferty, editors, *Language Modeling and Information Retrieval.* Academic Publishers, 2003.

[18] A. McCallum and B. Wellner. Conditional models of identity uncertainty with application to noun coreference. In *Proc. 17th NIPS*, 2004.

[19] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.

[20] V. Ng, S. Dasgupta, and S. M. N. Arifin. Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. In *Proc. 21st COLING/44th ACL*, pages 611–618, 2006.

[21] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proc. 21st SIGIR*, pages 275–281, 1998.

[22] A.-M. Popescu and O. Etzioni. Extracting product features and opinions from reviews. In *Proc. HLT/EMNLP*, 2005.

[23] S. Sarawagi. Information extraction. *Foundations and Trends in Databases*, 1(3):261–377, 2008.

[24] F. Song and W. B. Croft. A general language model for information retrieval. In *Proc. 22nd SIGIR*, pages 279–280, 1999.

[25] V. Stoyanov and C. Cardie. Topic identification for fine-grained opinion analysis. In *Proc. COLING*, 2008.

[26] J. Yi, T. Nasukawa, R. Bunescu, and W. Niblack. Sentiment analyzer: Extrating sentiments about a given topic. In *Proc. 3rd ICDM*, pages 427–434, 2003.

[27] C. Zhai. Statistical language models for information retrieval: A critical review. *Foundations and Trends in Information Retrieval*, 2(3):137–213, 2008.

[28] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM TOIS*, 22(2):179–214, 2004.