Matching Reviews to Objects using a Language Model

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

Abstract

We develop a general method to match unstructured text reviews to a structured list of objects. For this, we propose a language model for generating reviews that incorporates a description of objects and a generic review language model. This mixture model gives us a principled method to find, given a review, the object most likely to be the topic of the review. Extensive experiments and analysis on reviews from Yelp show that our language model-based method vastly outperforms traditional tf-idf-based methods.

1 Introduction

Consider a user searching for reviews of "Casablanca Moroccan Restaurant." The search engine would like to obtain as many reviews of this restaurant as possible, both to offer a high-quality result set for even obscure restaurants, and to enable advanced applications such as aggregation/summarization/categorization of reviews and recommendation of alternate restaurants. To solve this problem, it faces two high-level challenges: first, identify the restaurant review pages on the Web; and second, given a review, identify the restaurant that is being reviewed. There has been previous work addressing the first challenge (Section 2). We focus in this paper on the second.

The Web is replete with restaurant reviews available on top restaurant verticals such as Yelp and CitySearch, as well as newspaper articles, newsgroup discussions, blog posts, small local review aggregators and so forth. Ideally, the search engine would like to obtain reviews from all possible sources. While identifying the subject matter of a given review on the large sites may be amenable to structured extraction through wrapper induction or related techniques, it is typically not cost-effective to apply such techniques to smaller "tail" sites, and purely unstructured sources require alternate approaches altogether. In this paper, we explore the setting of matching reviews to objects using only their textual content. Note that matching reviews to objects is a pervasive problem beyond the restaurant domain. Shopping verticals like to aggregate camera reviews, entertainment verticals wish to collect movie reviews, and so on. We use restaurant reviews as a running example, but the techniques are general.

More specifically, the problem we consider in this paper is the following. Given a list of structured objects (restaurants/cameras/movies) and a text review, identify the object from the list that is the topic of the review. Our focus on textual content allows us to expand the universe of sources from which we can extract reviews to include sources that are purely textual, such as forum posts, blog posts, newsgroup postings, and the like. In fact, even among collections of "structured" sources like review aggregators, there are no highly accurate unsupervised techniques to match a known review page to an object. Structured (e.g., HTML) cues provide valuable leverage in attacking this problem, but the types of textual cues we focus on are also a key part of the puzzle; in such a context, our techniques can still contribute to the overall matching problem.

It is important to contrast our problem against two settings of related flavor — *entity matching*, whose goal is to find the correspondence between two structured objects and *information retrieval (IR)*, whose goal is to match unstructured short text (query) against unstructured text (document).

Our problem is considerably harder than entity matching for the following reasons. In matching two structured objects there is often a natural correspondence between their attributes, whereas no such correspondence exists between an object and

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 609–618,
Singapore, 6-7 August 2009.

its review. For instance, while trying to match a review to a restaurant object, it is unclear if a specific portion of the review refers to the name of the restaurant, or to its location, or is a statement not concerning specifics of the restaurant. Moreover, even if we wish to use entity matching, we must first recognize the entities from a review. There are two methods to do this, namely, wrapper induction and information extraction. Wrapper induction methods have serious limitations: they are applicable only to highly-structured websites and involve human labeling effort that is expensive and error-prone and entails constant maintenance to keep wrappers up-to-date. Information extraction methods (Cardie, 1997; Sarawagi, 2008), on the other hand, often have limited accuracy.

Our problem is also not amenable to classical IR methods such as tf-idf. For example, suppose we want to find the relevant restaurant for a given review. The standard tf-idf will treat the review as the query, the set of restaurant as documents and compute the tf-idf scores. Now consider a restaurant called "Food."[1] Since the term "food" is rare as a restaurant name, it will get a very high idf score and hence will likely be the top match for all reviews containing the word "food." In fact, unlike in traditional IR, a "query" (i.e., review) is long and a "document" (i.e., restaurant) is short — this demands adapting established IR concepts such as inverse document frequency and document length normalization to our setting. If we take the opposite view by considering reviews as documents and restaurants as queries, we still deviate from the IR setting, since now we need to rank and find the best "query" for a given "document." In Section 3.4, we illustrate the shortcomings of both these approaches.

In fact, the nature of the object database we consider provides several unique opportunities over traditional IR. First the "document", i.e., the object to be matched, has more semantics, since each document is associated with one or more semantic attribute, such as the name/location of the restaurant. Second, the "query", i.e., the text we are matching is known to be a review of the object, and hence is rendered in a language that is "review-like" — this can be modeled by a generative process that produces reviews from objects. Third, the set of objects we are interested in is given a priori, and we only seek to match reviews with one of these objects; this makes our problem more tractable than open-ended entity recognition.

Our contributions. We propose a general method to match reviews to objects. To this end, we postulate a language model for generating reviews. The intuition behind our model is simple and natural: when a review is written about an object, each word in the review is drawn either from a description of the object or from a generic review language that is independent of the object. This mixture model leads to a method to find, given a review, the object most likely to be the topic of the review.

Our method is light-weight and scalable and can be viewed as obviating the need for highly-expensive information extraction. Since the method is text-based and does not rely on any HTML structural clues, it is especially applicable to reviews present in blogs and the so-called tail web sites — web sites for which it is not feasible to maintain wrappers to automatically extract the object of a review.

We then report results on over 11K restaurant reviews from Yelp. The experiments and our extensive analysis show that our language model-based method significantly outperforms traditional tf-idf based methods, which fail to take full advantage of the properties that are specific to our setting.

2 Related work

Opinion topic identification is the work closest to ours. In a recent paper, Stoyanov and Cardie (2008) approach this problem by treating it as an exercise in topic coreference resolution. Though they have to deal with topic ambiguities and a lack of explicit topic mentions as in our case, their notion of a topic is not driven by a structured listing. There has been some work on fine-grained opinion extraction from reviews (Kobayashi et al., 2004; Yi et al., 2003; Popescu and Etzioni, 2005; Hu and Liu, 2004); see (Pang and Lee, 2008) for a comprehensive survey. Most of this body of work focused on identifying product features of the object under review, rather than identifying the product itself. Note that while a dictionary of products is often more readily available than a dictionary of product features, identifying objects of reviews is non-trivial even with the help of the former. Indeed, it has been reported that lexicon-

[1] 1569 Lexington Ave., New York, NY 10029. (212) 348-0200.

lookup methods have limited success on general non-product review texts (Stoyanov and Cardie, 2008). In general, this line of work is more rooted in the information extraction literature, where text spans covering the object (or features of the object) were extracted as the first step; in contrast, we do not have an explicit extraction phase. Since the (very extensive) list of candidate objects are given as input, our task is to rank all matching objects, and in this sense is closer in nature to information retrieval tasks. There has been some work on detecting reviews in large-scale collections (Ng et al., 2006; Barbosa et al., 2009); this is a logical step that precedes the review matching step, the topic of our paper.

Language modeling is becoming a powerful paradigm in the realm of information retrieval applications (Ponte and Croft, 1998; Hiemstra, 1998; Song and Croft, 1999; Lafferty and Zhai, 2003; Zhai, 2008). The basic theme behind language modeling is to first postulate a model for each document and for a given query select the document that is most likely to have generated the query; smoothing is an important means to manage data sparsity in language models (Zhai and Lafferty, 2004). As noted earlier, language models developed for IR are unsuitable for our setting. Furthermore, there are opportunities, such as the presence of structure in our data, which we use in this work (Section 3.2). In fact, in a subsequent paper, we show how a language model specific to each attribute can further improve the accuracy of review matching (Dalvi et al., 2009).

Entity matching is a well-studied topic in databases. There are several approaches to entity matching: non-relational approaches, which consider pairwise attribute similarities between entities (Newcombe et al., 1959; Fellegi and Sunter, 1969), relational approaches, which exploit the relationships that exist between entities (Ananthakrishna et al., 2002; Kalashnikov et al., 2005), and collective approaches, which exploit the relationship between various matching decisions, (Bhattacharya and Getoor, 2007; McCallum and Wellner, 2004). The EROCS system (Chakaravarthy et al., 2006), which uses information extraction and entity matching, is closest in spirit to our problem; they, however, employ tf-idf to match, which we show to be significantly sub-optimal in our setting.

3 Model and method

In this section we present the problem formulation, the basic generative model for reviews, a method based on this model to associate an object with a review, and the techniques to estimate the parameters of this model.

Problem formulation. Let $\mathcal{E}$ denote a set of objects. Each object $e \in \mathcal{E}$ has a set of attributes and let $\text{text}(e)$ denote the union of the textual content of all its attributes. Suppose we have a collection of reviews $\mathcal{R}$, where each review is written (mainly) about one of the objects in the listing $\mathcal{E}$. The problem now is to correctly associate each $r \in \mathcal{R}$ with exactly one of $e \in \mathcal{E}$.

We model each review as a bag of words. Therefore, notation such as "$w \in r$" for a word w and a review r makes sense. For a review r and an object e, let $r_e = r \cap \text{text}(e)$.

As a running example, we use $\mathcal{E}$ to denote the set of all restaurants and $\mathcal{R}$ to denote the set of all restaurant reviews.

3.1 A generative model for reviews

We first state the intuition behind our generative model: when a review r is written about an object e, some words in r (e.g., the name and the address of the restaurant) are drawn from $\text{text}(e)$ to refer to the object under discussion, while some other words are drawn from a *generic review language* independent of e.

Formally, let $\alpha \in (0, 1)$ be a parameter. Let $P_e(\cdot)$ denote a distribution whose support is $\text{text}(e)$; this corresponds to the distribution of words specific to the object e, taken from the description $\text{text}(e)$. We use $P_e(w)$ to denote the probability the word w is chosen according to this distribution. Let $P(\cdot)$ be an object-independent distribution whose support is the review language, i.e., all the words that can be used to write a review; we use $P(w)$ to denote the probability the word w is chosen according to this distribution. Now, for a given object e, a review r is generated as follows. Each word in r is generated independently: with probability α, a word w is chosen with probability $P_e(w)$ and with probability $1 - \alpha$, a word w is chosen with probability $P(w)$. Thus, the review generation process is a multinomial, where the underlying process is a mixture of object-specific language and a generic review language.

Given a review r and an object e, by our independence assumption,

$$\begin{aligned}\Pr[r \mid e] &= Z(r) \prod_{w \in r} \Pr[w \mid e] \\ &= Z(r) \prod_{w \in r} ((1-\alpha)P(w) + \alpha P_e(w)), \quad (1)\end{aligned}$$

where $Z(r)$ is a normalizing term that only depends on the length of r and the counts of the words in it. Recalling $r_e = r \cap \text{text}(e)$, we note that $P_e(w)$ assigns zero probability to $w \notin r_e$. From (1), we get

$$\begin{aligned}\Pr[r \mid e] &= Z(r) \prod_{w \in r \setminus r_e} (1-\alpha)P(w) \cdot \\ &\quad \prod_{w \in r_e} ((1-\alpha)P(w) + \alpha P_e(w)) \\ &= Z(r) \prod_{w \in r} (1-\alpha)P(w) \cdot \\ &\quad \prod_{w \in r_e} \left(1 + \frac{\alpha}{1-\alpha} \frac{P_e(w)}{P(w)}\right). \quad (2)\end{aligned}$$

Note that Eq. (2) appears similar to the formula obtained in the language model approach for IR (Hiemstra and Kraaij, 1998); the interpretation of terms, however, is very different. For instance, $P(w)$ in our case is computed over the "query" corpus whereas the analogous term (collection frequency) in (Hiemstra and Kraaij, 1998) is computed over the "document" corpus. As the "Food" restaurant example in Section 1 suggests, using the "document" frequency is undesirable. The use of "query" corpus frequency arises naturally from our generative story and also guides us to a different way to estimate $P(w)$; see Section 3.3.

3.2 Matching a review to an object

Given the above review language model (RLM), we now state how to match a given review to an object. According to our model, the most likely object e^* to have generated a review r is given by

$$e^* = \arg\max_e \Pr[e \mid r] = \arg\max_e \frac{\Pr[e]}{\Pr[r]} \cdot \Pr[r \mid e].$$

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

$$e^* = \arg\max_e \Pr[r \mid e],$$

or equivalently,

$$e^* = \arg\max_e \log \Pr[r \mid e].$$

Since $Z(r) \prod_{w \in r}((1-\alpha)P(w))$ is independent of e, using (2), we have

$$e^* = \arg\max_e \sum_{w \in r_e} \log \left(1 + \frac{\alpha}{1-\alpha} \frac{P_e(w)}{P(w)}\right). \quad (3)$$

3.3 Estimating the parameters

We now describe how to estimate the parameters of the model, namely, $P(\cdot)$, $P_e(\cdot)$, and α.

Recall that $P(\cdot)$ is the distribution of generic review language. Ideally, for each review r, if we know the component $r^{(e)}$ that came from the distribution $P_e(\cdot)$ and the component $r^{(g)}$ that came from $P(\cdot)$, then we can collect the $r^{(g)}$ components of all the reviews in $\mathcal{R}$, denoted as $\mathcal{R}^{(g)}$, and estimate $P(\cdot)$ by the fraction of occurrences of w in $\mathcal{R}^{(g)}$. More specifically, let $c(w, \mathcal{R}^{(g)})$ denote the number of times w occurs in $\mathcal{R}^{(g)}$. With add-one smoothing, we estimate

$$P(w) = \frac{c(w, \mathcal{R}^{(g)}) + 1}{\sum_{w'} c(w', \mathcal{R}^{(g)}) + |V|},$$

where $|V|$ is the vocabulary size.

In reality, we only have access to r and not to the components $r^{(e)}$ and $r^{(g)}$. If we have an aligned review corpus $\mathcal{R}'$, where for each review r, we know the true object e that generated it, we can closely approximate $r^{(e)}$ with r_e.[2] Let no-obj($\mathcal{R}'$) be the set of processed reviews where for each review-object pair (r, e), words in $\text{text}(e)$ are removed from r. By treating no-obj($\mathcal{R}'$) as an approximation of $\mathcal{R}^{(g)}$, we can compute $P(w)$ in the aforementioned manner. If we only have access to a review collection $\mathcal{R}'$ with no object alignment, there are other ways to effectively approximate $\mathcal{R}^{(g)}$; see Section 5.3 for more details.

Unlike $P(\cdot)$, we cannot learn an individual language model $P_e(\cdot)$ for each e, since we cannot expect to have training examples of reviews for each possible object e in the dataset. Thus, we need a simpler way to model $P_e(w)$. The most naive way would be to assume a uniform distribution, i.e., $P_e(w) = 1/|\text{text}(e)|$. However, each word

[2] There can be exceptions to this, e.g., review of a restaurant called "Tasty Bites" might use the word "tasty" from the review language, but not to refer to the restaurant. Nonetheless, we believe these will be rare exceptions and will not have significant effect in the estimation of $P(\cdot)$.

in $\text{text}(e)$ may not be generated with equal probability. In our running example, consider the case when $\text{text}(e)$ contains the full name of the restaurant, i.e., "Casablanca Moroccan Restaurant." A review for this restaurant is more likely to choose the word "Casablanca" than any other word to refer to this restaurant since this is arguably more informative than "Moroccan" or "Restaurant." This can be captured by using the frequency f_w of the word w in $\mathcal{R}$ or in $\{\text{text}(e) \mid e \in \mathcal{E}\}$. For a suitable function $g(w)$ that is inversely growing as f_w (say, $g(w) = \log(1/f_w)$), we let

$$P_e(w) = \frac{g(w)}{\sum_{w' \in \text{text}(e)} g(w')}.$$

Alternatively, it is possible to construct models where $P_e(w)$ is more directly estimated from the data; in fact, one can also use suitable translation models to estimate $P_e(w)$ for w that may not even occur in $\text{text}(e)$ — this will help in cases where reviews use an abbreviation such as "Casa" or "CMR" to refer to our running example. Such models require either fine-grained labeled examples or, as we show in (Dalvi et al., 2009), more sophisticated estimation techniques.

It is tempting to assume that common words such as "Restaurant" may not contribute towards matching a review to an object and hence one can conveniently set $P_e(w) = 0$ for such words w. (Such a list of words can easily be compiled using a domain-specific stopword list.) This may hurt — in our example, the presence of the word "Restaurant" in a review might help to disambiguate the object of reference, if the listing were also to contain a "Casablanca Moroccan Cafe".

3.4 Properties of the model

Eq. (3) indicates that our method (denoted as RLM) gives less importance to common words with high $P(w)$. This corresponds to the intuition behind the standard tf-idf scheme. Why, then, do we expect RLM to be more effective? Here, we discuss the salient features of our method, contrasting it with tf-idf in particular.

First, we take a closer look at different ways to apply tf-idf techniques to our setting. Since the task is to find the most relevant object given a review, a naive way to apply the standard tf-idf (denoted TFIDF) will treat each review to be the query and each object to be a document and score documents using the standard tf-idf scoring. This, however, leads to severe problems since this computes the inverse document scores over the object corpus — recall the "Food" example in Section 1.

A more reasonable way to apply tf-idf is to instead treat objects as queries and reviews as documents for computing tf-idf scores (denoted TFIDF$^+$). For a word w, let $Q(w) = \frac{\text{df}(w)}{N}$, where N is the number of reviews in the corpus and $\text{df}(w)$ is the number of reviews containing w. Given a review r and an object e, the score of the object is given by $\sum_{w \in r_e} \log(1/Q(w))$, and we want to pick the object with the maximum score. As we will discuss later, document-length normalization (i.e., normalizing by object description length so that a restaurant with a long name does not get an unfair disadvantage) is still non-trivial here.

As noted earlier, Eq. (3), used by RLM for matching reviews with objects, has a striking resemblance to the TFIDF$^+$ scoring function. Both have the form

$$e^* = \arg\max_e \sum_{w \in r_e} \log f(w),$$

where for RLM,

$$f(w) = f_R(w) = 1 + \frac{\alpha}{1-\alpha}\frac{P_e(w)}{P(w)},$$

and for TFIDF$^+$,

$$f(w) = f_B(w) = \frac{1}{Q(w)}.$$

In both cases, $f(w)$ is monotonically decreasing in the frequency of w in the corpus. However, there are several differences between the two cases. We highlight some of them here, with the aim of illustrating the power of our review language model (RLM).

Object length normalization. First note that the $P_e(w)$ term in $f_R(w)$ acts as an object length normalizing term, i.e., it adds up to one for each e and weighs down $P(w)$ for objects with long $\text{text}(e)$. This also has the effect of penalizing reviews that are missing critical words in the object description. In contrast, $f_B(w)$ is unnormalized with respect to the object length. The standard document normalization techniques in IR do not apply well to our setting since our "documents" (i.e., object descriptions) are short. E.g., if the object description contains only one token, the standard cosine-normalization technique (Salton et al.,

1975) will yield a normalized score of 1 *irrespective* of the token. Thus for a review containing the words "Food" and "Casablanca", the standard normalization will yield the same score for a restaurant named "Food" and a restaurant named "Casablanca", ignoring the fact that "Food" is much more likely to be an object-independent term. Note that this only becomes a problem when the entire "document" is part of the match, which rarely happens in an IR setting where the documents are typically much longer than the queries. Indeed, in our experiments, we observe lower performance when we apply cosine-normalization to the tf-idf scores. On the other hand, in $f_R(w)$, the $P(w)$ term can still distinguish the two aforementioned objects even when $P_e(w)$ are equal.

Dampening. With $\alpha < 1$, $f_R(w)$ is effectively a dampened version of $\frac{P_e(w)}{P(w)}$. In other words, differences between very frequent words and very infrequent words are somewhat smoothed out. Indeed, if we modify TFIDF$^+$ by introducing a similar dampening factor into $f_B(w)$, we observe improvement in its performance (Section 5.4).

Removing mentions of an object. Another difference is that in RLM, $P(w)$ is estimated on reviews with object mentions removed, since the model indicate that $P(w)$ accounts for object-independent review language. In contrast, TFIDF$^+$ computes $Q(w)$ on full reviews. We illustrate the difference on the following example. Consider a review that reads "...Maggiano's has great Fondue." If "Maggiano's" and "Fondue" both occur the same number of times in the corpus, then they get the same idf (i.e., $Q(w)$) score. In RLM, however, "Maggiano's" will get much smaller probability in the generic review distribution $P(\cdot)$ than "Fondue", since "Maggiano's" almost always occurs in reviews as restaurant name mentions, thus is removed from the estimation of its $P(\cdot)$ probability. On the other hand, the word "Fondue" is more likely to retain higher probability in $P(\cdot)$ since it tends to appear as dish names. As a result, our model will assign higher weight to "Maggiano's Restaurant" than "Fondue Restaurant". As we can see, RLM evaluates the ability of a word to identify the review object rather than rely on the absolute rarity of the word, which is done by tf-idf.

Using term counts. One last difference is that $f_R(w)$ uses term counts of words rather than the standard document counts used by $f_B(w)$. Our evaluation suggests that at least in practice, this does not have a big impact on the overall accuracy.

In the experiments we show that these factors together account for the performance difference between RLM and tf-idf. Our model gives a principled way to introduce these factors, however.

4 Data

In this section we describe the dataset constructed for the task of matching restaurant reviews to the corresponding restaurant objects. Our goal is to obtain a large collection of reviews on which to estimate the generic language model, with a significant portion of them aligned with the objects for which the reviews were written; this portion will serve as the gold-standard test set.

To this end, we obtained a set of reviews from the Yelp website, `yelp.com`. This website contains a collection of reviews about various businesses and for each business, has a webpage containing the business information and a list of reviews. We crawled all restaurant pages from Yelp. For each restaurant, we extracted its name and city location from the business information section via HTML cues, and a list of no more than 40 reviews. We obtained the textual content of 299,762 reviews, each aligned with one of a set of 12,408 unique restaurants hosted on Yelp. Note that while our technique is not targeted for head sites like Yelp (where wrapper induction might be a more accurate approach), this provides a large-scale dataset, conveniently labeled with object information, and simulates the tail-site scenario where we rely heavily on the textual content of reviews to identify objects.

Many of the reviews in Yelp do not contain any identifying information. In fact, some of them are as short as "Great place. Awesome food!!". We processed the dataset to retain only reviews that mention the name of the restaurant, even if partially, and, when the restaurant name is a common word, also the city of the restaurant. Each of the remaining reviews is expected to have enough information for a human to identify the restaurant corresponding to the review.

To further increase the difficulty of the matching task, we obtained a much more extensive list of restaurant objects in the Yahoo! Local database, which contains 681,320 restaurants. Our task is to match a given Yelp review, using only its free-form textual content, with its corresponding

restaurant in the Yahoo! Local database. We then proceeded to generate the gold standard that contains the correct restaurant in the Yahoo! Local database for each review. We employed geocoding to match addresses across the two databases along with approximate name matches. Note that in the final dataset, only half of the restaurants have the exact same name listed in both Yelp and Yahoo! Local; this limits the success of naive dictionary-based methods.

The final aligned dataset contained 24,910 Yelp reviews ($\mathcal{R}$), covering 6,010 restaurants. We set aside half of the reviews ($\mathcal{R}'$) to estimate the models and the other half ($\mathcal{R}_{\text{test}}$) to evaluate our technique. We also set aside 1,000 reviews as development set, on which we conducted initial experiments. The total size of the test corpus, $\mathcal{R}_{\text{test}}$ was 11,217. The splitting of $\mathcal{R}$ into $\mathcal{R}'$, $\mathcal{R}_{\text{test}}$, and the development set was done in such a way that there are no overlapping restaurants between them. Also, the reviews that were filtered out because of lack of identifying information were added back to $\mathcal{R}'$ for learning the review language model, expanding $\mathcal{R}'$ to a total of 205,447 reviews.

5 Evaluation

In this section we evaluate our proposed review-language based matching algorithm RLM.

5.1 Experimental considerations

Baseline system. We use the TFIDF and TFIDF$^+$ algorithms described in Section 3.4 as baseline algorithms. Since we are comparing objects that can have varying lengths, we tried the standard cosine-normalization techniques for document length normalization. For reasons described in Section 3.4, however, the normalization significantly lowered the accuracy. All the numbers reported here are using tf-idf scores without normalization.

Efficiency. For both RLM and the baseline algorithms, it is impractical to compute the similarity of a review with each object in the database. Since all objects that do not intersect with the review have a zero score, we built an inverted index to retrieve all objects containing a given word. Even a simple inverted index can be very inefficient since for each review, words such as "Restaurant" or "Cafe" retrieve a substantial fraction of the whole database. Hence, we further optimized the index by looking at the document frequencies of the words and considering word bigrams in object descriptions. The index only retrieves objects that have a non-trivial overlap with the review; e.g., an overlap of "Casablanca" is considered non-trivial while an overlap of "Restaurant" is considered trivial. Once these candidates are retrieved, our scoring function takes into account all overlapping tokens.

For the YELP dataset, the index returns an average of 200 restaurants for each review. This points to the general difficulty of review matching over a large corpus of objects, since a simple dictionary-based named-entity recognition will hit at least 200 objects for many reviews.

Experiment settings. For RLM, we conducted initial experiments and performed parameter estimation on the development data. The experimental settings we used for RLM are as follows: we set $g(w) = \log(1/f_w)$ for P_e, where f_w is estimated on the review collection. $P(w)$ is estimated on all reviews in R', where for each review, all tokens of its corresponding $\text{text}(e)$, if present, are removed, in order to approximate the *generic* review language independent of e, as required by our generative model. We estimate α to be 0.002, tuned on the development set; in our experiments, we observe that the performance is not very sensitive to α.

5.2 Main results

In this section we present the main comparisons between RLM and the baseline in details.

Performance measure. Our task resembles a standard IR task in that our algorithm ranks candidate objects for a given review by their "aboutness" level. Unlike a standard IR task, however, we are not interested in retrieving multiple "relevant" objects, as each review in our dataset has only one single correct match from $\mathcal{E}$. A review match is correct if the top-1 prediction (i.e., e^*) is accurate. In what follows, we report the average accuracy for various experimental settings. Note that we can take the average accuracy over all reviews (reported as micro-average), regardless of which restaurants they are about; or we can first compute the average for reviews about the same restaurant, and report the average over all restaurants (macro-average). When not specified, we report the micro-average.

Main comparisons. Table 1(a) summarizes the main comparison. Our proposed algorithm RLM

Method	Micro-avg.	Macro-avg.
RLM	0.647	0.576
TFIDF$^+$	0.518	0.481
TFIDF	0.314	0.317

(a) Main comparison.

Method	Micro-avg.	Macro-avg.
RLM-UNIFORM	0.634	0.562
RLM-UNCUT	0.627	0.546
RLM-DECAP	0.640	0.573

(b) RLM variants.

Method	Micro-avg.	Macro-avg.
TFIDF$^+$-N	0.586	0.523
TFIDF$^+$-D	0.593	0.533
TFIDF$^+$-O	0.522	0.488
TFIDF$^+$-ND	0.628	0.549
TFIDF$^+$-NDO	0.647	0.576

(c) TFIDF$^+$ variants.

Table 1: Average accuracy of the top-1 prediction for various techniques. Micro-average computed over 11,217 reviews in $\mathcal{R}_{\text{test}}$; macro-average computed over 2,810 unique restaurants in $\mathcal{R}_{\text{test}}$.

clearly outperforms the TFIDF$^+$ baseline measured by either micro- or macro-average accuracy. The standard TFIDF, as predicted, performs the worst.

Some reviews can be particularly difficult to match, which can be reflected in a low matching score. Nonetheless, we predict the most likely object. Suppose we impose a threshold and return the most likely object only when its score is above threshold, we can then compute precision and recall at different thresholds. Figure 1 presents the precision–recall curve (using micro-average) for both RLM and TFIDF$^+$. Again, RLM clearly outperforms TFIDF$^+$ across the board.

We then generalize the definition of accuracy into accuracy@k: a review is considered as correctly matched if one of the top-k objects returned is the correct match. We plot accuracy@k as a function of k. While the gap between RLM and TFIDF$^+$ is smaller as k increases, RLM clearly outperforms TFIDF$^+$ for all $k \in \{1, \ldots, 10\}$.

One final comparison is accuracy@1 as a function of the review length. Given our current setting, longer reviews might be more difficult to match since they may include more proper nouns such as dish names and related restaurants, and

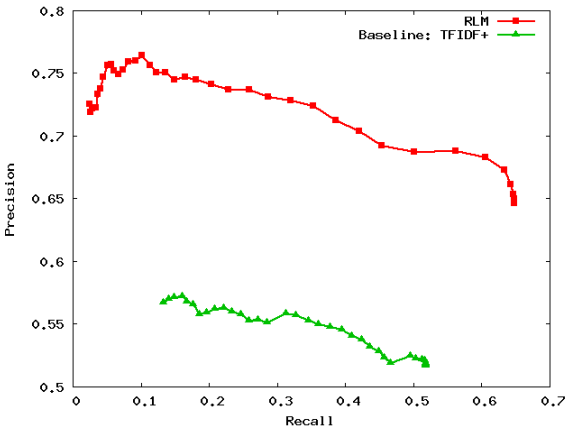

Figure 1: Precision–recall curve (of top one prediction): RLM vs. TFIDF$^+$ baseline.

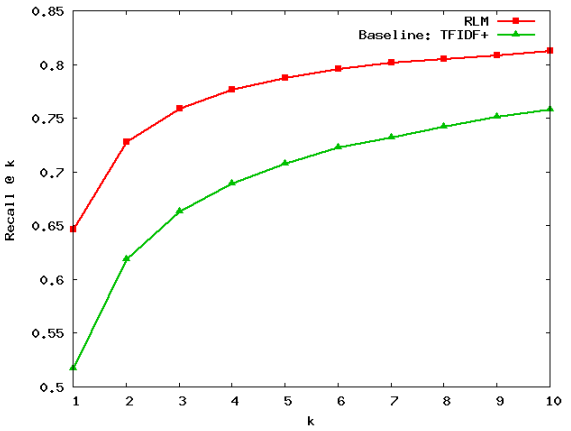

Figure 2: Accuracy@k (percentage of reviews whose correct match is returned in one of its top-k predictions): RLM vs. TFIDF$^+$ baseline.

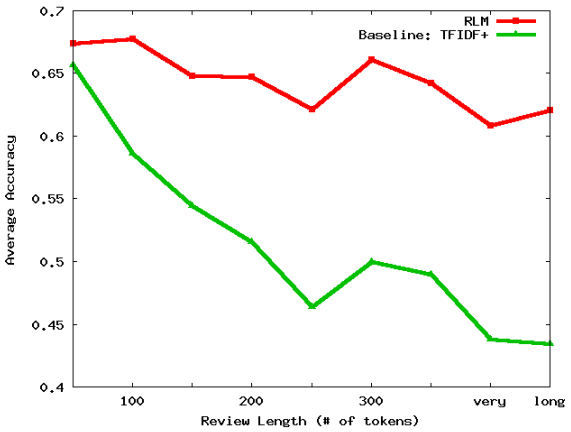

Figure 3: Average accuracy of the top-1 prediction for reviews with different length (on test set): RLM vs. TFIDF$^+$ baseline.

yield a longer list of highly competitive candidate objects. Interestingly, the gap between RLM and TFIDF$^+$ is much smaller for shorter reviews. As reviews get longer, the performance of RLM is relatively stable, whereas the performance of TFIDF$^+$ drops down significantly.

5.3 Experimental choices for RLM

We now examine the experimental choices we made for different components of RLM by defining the following variations of RLM.

RLM-UNIFORM: rather than setting $g(w) = \log(1/f_w)$ for P_e, we use the uniform distribution $P_e(w) = 1/|\text{text}(e)|$. From the third line of Table 1 (b), there is a slight accuracy drop of $\sim 1.3\%$.

RLM-UNCUT: suppose we only have access to a review corpus with no alignment to $\text{text}(e)$, and thus have to approximate $P(w)$ by estimating it on the set of original "un-cut" reviews, how much does that affect our performance? As indicated in the fourth row of Table 1 (b), this reduces accuracy by about 2% on our test data.

RLM-DECAP: as an alternative way to deal with lack of aligned data, we consider a variation of the above algorithm by removing all the capitalized words from un-annotated reviews. Clearly, this can result in both "over-cutting" and "under-cutting" of true restaurant name mentions. However, as indicated in the fourth row of Table 1 (b), this is very close to the best accuracy achieved. Thus, an effective model can be learned even without aligned data.

5.4 Revisiting TFIDF$^+$: what's amiss?

In this section we revisit the main differences between our model and the TFIDF$^+$ outlined in Section 3.4, and investigate their empirical importance by introducing these features into TFIDF$^+$ and examine their effectiveness in that framework.

Object length normalization. We consider a modified TFIDF$^+$ measure $f_M(w) = P_e(w)/Q(w)$, which we call TFIDF$^+$-N (normalized). As shown in Table 1 (c), this change alone can increase the average accuracy by nearly 7%.

Dampening. We consider a modified TFIDF$^+$ measure $f_M(w) = 1 + \beta \cdot \frac{N}{\text{df}(w)}$, which we call TFIDF$^+$-D. Table 1 (c) reports the performance of using this measure, with $\beta = 0.1$ (set on development data). Again, this measure alone can induce over 7% increase in accuracy. Indeed, combining normalization and dampening, (i.e., $f_M(w) = 1 + \beta \cdot P_e(w) \cdot \frac{N}{\text{df}(w)}$), denoted as TFIDF$^+$-ND, we get comparable performance to RLM-UNCUT.

Removing mentions of objects. Again, we can incorporate this in a heuristic way in TFIDF$^+$, which we denote by TFIDF$^+$-O. Interestingly, while using the original $f_B(w)$ function with $\text{df}(w)$ computed on the object-removed review collection does not yield a big improvement, this does bring the performance of the fully modified TFIDF$^+$ to the same level of the standard RLM (see line marked TFIDF$^+$-NDO.)

Using term counts. Our investigation suggests that at least in practice, using $Q(w)$ vs. $P(w)$ is not a critical decision, as a fully modified TFIDF$^+$ can achieve the same performance using $\text{df}(w)$ to quantify frequency of the word. Our experiments on this dataset show that each of the other modeling decisions incorporated in RLM is important.

6 Conclusions

We proposed a generative model for reviews where reviews are generated from the mixture of a distribution involving object terms and a generic review language model. The model provides us a principled way to match reviews to objects. Our evaluation on a real-world dataset shows that our techniques vastly outperforms standard tf-idf based techniques.

Acknowledgments

We thank Don Metzler for many discussions and the anonymous reviewers for their comments.

References

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

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

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

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

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

N. Dalvi, R. Kumar, B. Pang, and A. Tomkins. 2009. A translation model for matching reviews to objects. Manuscript.

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

D. Hiemstra and W. Kraaij. 1998. Twenty-one at TREC7: Ad-hoc and cross-language track. In *Proc. 7th TREC*, pages 174–185.

D. Hiemstra. 1998. A linguistically motivated probabilistic model of information retrieval. In *Proc. ECDL*, pages 569–584.

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

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

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

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

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

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

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

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

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

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

G. Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

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

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

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

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

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

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