Object Matching in Tweets with Spatial Models

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

ABSTRACT

Despite their 140-character limitation, tweets embody a lot of valuable information, especially temporal and spatial. In this paper we study the geographic aspects of tweets, for a given object domain. We propose a user-level model for spatial encoding in tweets that goes beyond the explicit geocoding or place name mentions; this model can be used to match objects to tweets. We illustrate our model and methodology using restaurant websites as the objects, and show a significant improvement in performance over using standard language models. En route, we obtain a method to geolocate users who tweet about geolocated objects; this may be of independent interest.

Categories and Subject Descriptors. H.3.m [Information Storage and Retrieval]: Miscellaneous

General Terms. Algorithms, Experimentation, Measurements

Keywords. Tweets, Spatial model, Language model, Object matching

1. INTRODUCTION

Tweets are steadily being treated as information sources and there is a growing body of research work to understand and use them in web applications. Tweets provide information that, in a sense, is complementary to what other web sources provide. For example, a user may review a restaurant only once, but may tweet about the same restaurant multiple times. A particular Twitter application that is becoming popular is topic- and object-recognition: identify which object the tweet is about. The broad success of tweetbeat.com during the 2010 World Cup Soccer, where the aboutness of tweets had to be determined, is a testament to the importance of this application. Motivated by applications such as these, in this paper we study the object matching problem in tweets.

Most of the research so far has been to study individual tweets from an information retrieval and content analysis point of view, to use the set of tweets from a user for recommendation purposes, and to get insights into the social processes behind tweeting. The 140-character limitation on tweets makes many of these tasks already challenging: there is hardly any room to provide complete information about anything, even if the author of the tweet so desired. Given that many of the tweets are content-free,1 there is a real limitation on how far tweets by themselves can be used for interesting applications such as object matching.

Fortunately, there are two redeeming aspects to tweets: their volume and the context in which they are generated. By context, we mean the metadata about the author of the tweet and the spatial/temporal information associated with each tweet. A judicious filtering of the content-free tweets combined with a careful use of its context opens up new possibilities for using the information tweets contain.

Geographic information is a vital component of the context of a tweet. More than one-third of all Twitter users have a valid geographical location and more than half of them have a geographical location that is at a city level; of course, these are based on self-reporting and hence could be erroneous. (More on these numbers in Section 3.2.) And, 3–4% of all tweets have a latitude-longitude (lat-lon) associated with them. This fraction will grow with the proliferation of GPS-enabled devices and social applications such as Foursquare or Gowalla.2 Furthermore, even when such information is not explicitly declared, it can be possible to infer it from the location of places a user typically tweets about. All of these raise the possibility of using geographic information in an intrinsic manner in order to improve the usability of tweets as information.

In this paper we consider a genre of applications, where the goal is to match a tweet to an object, if possible, where the object is from a list of objects in a given domain. We assume that the geographic location of the entities is known. For this object matching task, we develop a user model that utilizes geographic information in an intrinsic way: the probability of a user tweeting about the object depends on the distance between the user’s location and the object’s location. We illustrate our model and method by using restaurants as the object domain. There have been a few recent attempts to incorporate geography into topic modeling and

1A Pear Analytics study from 2009 (http://www.pearanalytics.com/blog/wp-content/uploads/2010/05/Twitter-Study-August-2009.pdf) argues that over 40% of the tweets are small talk.

2According to http://www.pewinternet.org/~/media/Files/Reports/2010/PIP-Location-based-services.pdf, on each day, 1% of the online population uses these services.
related tasks but none of these is particularly suitable for our application (more on this in Section 2).

Main contributions. We formulate a simple user model for generating a tweet about an object. In our model, there are three quantities: each user is associated with an interest in the set of objects (say, restaurants) that captures the propensity of the user to tweet about an object from the set, each object has an associated popularity, and each user and each object have their own geographic locations (locations of objects are given as input to the model, whereas locations of users need to be inferred). Now, the probability that the user tweets about an object is a product of the user’s interest in the object, the popularity of the object, and a decreasing function of the distance between the user and object locations.

With this model, we show an estimation procedure based on the EM algorithm. Unfortunately, the estimation can be computationally challenging. We then show that under reasonable simplifying assumptions on the distribution of users and objects, a computationally efficient method can be used to learn the parameters of the model. As a byproduct, we also obtain a principled method to geolocate tweets: infer the location of a user based on the tweets about geolocated objects.

Our experiments on a large-scale collection of tweets that show our geography-enabled model is able to achieve significant performance gains over geography-less models. We also show that our method to infer geolocation of the users performs well in practice: in our restaurant-related dataset, for users with three or more tweets, the median error between the user’s self-reported location and the location inferred by our method is around 10 miles.

Organization. In Section 2, we discuss the related work on topic models that incorporate geography, matching objects in short user-generated content, and geolocating user-generated content. In Section 3, we argue why our problem is hard and why traditional methods will not be applicable. In Section 4, we present the basic model that consists of a distance component based on the user model described above and a language model component that models textual information. In Section 5, we develop a method to estimate the parameters of the model in a principled manner; here, we make certain simplifying assumptions that make the estimation procedure computationally tractable so that it can be applied to a large domain such as tweets. In Section 6, we describe the data source and its properties. In Section 7, we examine properties of our distance model and in Section 8, we study the performance of the full model on the matching task, focusing on the performance improvements enabled by the combination of the distance model with the language model. Concluding remarks are presented in Section 9.

2. RELATED WORK

The line of work closest to ours is that of topic models that incorporate geography. O’Connor et al. study the geographic linguistic variation in tweets [8, 11]. They propose a geographic topic model that incorporates regional vocabulary variations in the way the tweet text is generated. They use the geotagged tweets in their study and their model can be used to classify a tweet into a geographic region. Extending topic modeling to incorporate geographic information has been recently addressed in general information retrieval and NLP contexts. Mei et al. [10] postulate a model in which the distribution of topics is conditioned on geographical location; however they assume that the set of locations is given. Sizov [14] combines text and geographic information, where each topic generates latitude and longitude from two topic-specific Gaussian distributions. He shows that this improves tasks such as tag recommendation and clustering. Yin et al. [17] also study a similar problem but the geographic topic regions are discovered in a more systematic manner. Our work differs from these in the following ways: their granularities are much coarser than what we can afford, they do not incorporate the location of a user, and their goal is not to do object matching.

The question of matching objects to short text such as reviews and blogs was recently addressed in [6, 5], where simple translation models were proposed for the matching; such models served as motivation behind our language model. These papers, however, do not attempt to model the geographic aspects of the data, which is unique to our setting.

The geographic information that is present in tweets and in other user-generated content has lent itself to many interesting data mining studies. Backstrom et al. [1] studied the role of geography in search engine queries. Crandall et al. [4] studied similar questions for geotagged images. They used text tags of images and geospatial data to find out the location of photos; see also the work of Rattenbury et al. [12] on Flickr tags. Backstrom et al. [2] proposed a method to infer the geographic location of users based on the stated locations of their social connections; their method is based on postulating a friendship creation model that resembles the geographic component of our model. Recently, Wing and Baldridge [15] proposed a text-based supervised method for geolocating tweets; their method obtains a median error of 479km. All these papers focus only on locating an object in the space using several clues about them. We, on the other hand, make this location detection as part of our model, with the end goal of matching entities to tweets. Finin et al. [9] tried to use crowd-sourcing in order to extract named entities in tweets; they do not use any geographic information. Tweets are being studied from a sociological point of view: there is a growing literature on this and we refer the readers to [3, 16, 13] and the references therein.

3. A CLOSER LOOK AT THE PROBLEM

In this section, we discuss motivations of our modeling choices.

3.1 Is text enough?

We can cast our problem as matching unstructured text such as tweets with structured entities in a database listing (e.g., restaurants with attributes like name, address, and cuisine). This can be seen as a cross of two problems: entity matching in databases, which matches pairs of structured entities, and document retrieval in IR, which matches unstructured queries with unstructured documents. Previous work [5, 6] has looked into matching unstructured text reviews with structured entities. The main idea behind these

\[^3\]As we will see in Section 7, our geolocation method has a median error of below 10 miles; however, these results are not comparable since [15] use all tweets of the user whereas we focus only on restaurant-related tweets.
of user locations can immensely help us in disambiguating between the matching restaurants.

As an example, suppose we know the location of the user in the tweet above to be Somerville, MA. Then, we can match the tweet to Ronnarong in Somerville with higher confidence than Ronnarong in other locations. This can also help us further lower the confidence over Yum Thai and Thai Tapas, if those restaurants are not near Somerville.

3.2 Can users be geolocated?

In the previous section, we argued that knowledge over user locations can be very helpful for the matching task. Is it trivial to identify obtain this information at a useful granularity? One obvious source seems to be the profile information, which is part of the meta-information accompanying each tweet. However, not all users report their locations in their profiles. Based on an analysis of one month of tweet feed (June, 2010), out of the 22 million unique users who tweeted in this period, 44% of the users did not specify location in their profile. This accounted for 19.4% of the total tweet traffic.

Among the users who do have non-empty location field, not all of the specified location values correspond to valid geographical locations, e.g., Next to you. We applied the Yahoo! Geolocation API, and obtained the country and city information from the profile locations when the API was able to recognize one. We found that overall, 33% of the profiles contain recognizable country names, and only 20% (accounting 25% of the tweet traffic) contain city names. There are also twitter clients that let users with a geo-enabled device to set an exact latitude and longitude as their location, but they amount to less than 2% of users. If we were to depend on the self-reported locations in user profiles, clearly we miss out on 80% of the users (75% of the tweets), not to mention tweets where the self-reported locations do not reflect the actual place the user resides in.

In this work we take a different approach. Instead of relying on self-reported locations, we try to infer the location of a user based on the geo-locations of entities the user tweets about. This presents a technical challenge: we need the user location to compute accurate matches, and we need correct matches to infer the location. Our approach is to use a probabilistic model that combines the language model with the distance model, and jointly infer the location of the users and the matches between tweets and entities. We experimentally demonstrate that our model is very effective in inferring user locations, and for many of the users we can identify the location to within 10 miles. Thus, our technique for location inference helps even for users with valid reported locations. E.g., we might identify that a user with a reported location in New York is in fact in Lower Manhattan. In the next section, we describe our joint model.

4. MODEL

Let \( U \) be a set of users, where each user \( u \in U \) has a location \( \ell(u) \). Let \( E \) be a set of entities, where each entity \( e \in E \) has a set of attributes including its location \( \ell(e) \). Finally, let \( T \) be a set of tweets, where each tweet \( t \in T \) it does not help as it is not part of the address we have in our database.

\[3\] Since this API does not work with foreign text, we removed those profile location that contained non-ASCII characters.
consists of a piece of text, and is associated with a user $u \in U$, and needs to be matched to an entity $e \in E$.

We assume there is an underlying distribution $P$ over the set $T \times U \times E$. Our goal is to model $P$. This will enable us to compute $P(e \mid t, u)$ for a triple $(t, u, e)$ drawn from this distribution, so that we can predict the best $e^*$ for an observed tweet–user pair $(t, u)$.

There are two components of the probabilistic model.

(i) The distance model captures the relationship between $U$ and $E$. It is given by $P(e, u)$, which captures the propensity of a user $u$ tweeting about an entity $e$.

(ii) The language model captures the relationship between $E$ and $T$. It is given by $P(t \mid e, u)$, which captures the distribution of words in a tweet $t$ corresponding to a user $u$ and an entity $e$.

The complete model that captures the probability distribution over $T \times U \times E$ can be expressed as

$$P(e, t, u) = P(t \mid e, u)P(e, u).$$  \hspace{1cm} (1)

We now discuss the distance model (Section 4.1) and the language model (Section 4.2) in more detail.

### 4.1 Distance model

In this section we present our model for $P(e, u)$, which captures the propensity of a user $u$ tweeting about an entity. We expect the geographical distance between a user $u$ and an entity $e$ to play an important role in this model, and use $d(e, u)$ to denote the distance between $\ell(u)$ and $\ell(e)$.

We assume that each user $u$ is associated with an interest $\alpha(u)$ with respect to the entity set $E$, e.g., if $E$ is the set of restaurants, then $\alpha(u)$ will capture the interest of the user $u$ in tweeting about restaurants. Similarly, we assume that each entity $e$ has a popularity $\beta(e)$. Finally, the probability of observing a pair $(e, u)$ is directly proportional to the interest of the user and the popularity of the entity and monotonically decreases with the distance between them. Assuming a polynomial decay, we have

$$P(e, u) \propto \alpha(u)\beta(e)(d_0 + d(e, u))^{-k},$$  \hspace{1cm} (2)

where $d_0$ and $k$ are parameters that govern how fast the probability drops with distance.

We can normalize the above equation by summing over all pairs of entities and users. Let

$$g(e, u) = \alpha(u)\beta(e)(d_0 + d(e, u))^{-k},$$  \hspace{1cm} (3)

we have

$$P(e, u) = \frac{g(e, u)}{\sum_{(e', u') \in E \times U} g(e', u')}.$$  \hspace{1cm} (4)

Note that we use a distance function that exhibits a polynomial decay with respect to the distance between the user and the entity. Alternatively, we could have used, say, an exponentially decaying function. Here, we give the theoretical rationale for our distance function; Section 7 provides more empirical evidence for our choice.

In theory, exponentially decaying function does not lead to intuitive properties that we expect. For instance, suppose we have a model where the probability of a user $u$ visiting an entity $e$ is proportional to $e^{-kd(e, u)}$. Consider the case when we have a user $u$ and two entities $e_1$ and $e_2$ that are $d_1$ and $d_2$ away from $u$. Suppose they are all located on the same line, with $d_2 = d_1 + y$. If $y$ is a constant $\frac{\log 2}{k}$, we have $P(e_1, u) = e^{-k(d_1 + d_2)} = e^{k y} = 2$. That is, the user is twice as likely to visit $e_1$ than $e_2$, irrespective of $d_1$. This behavior is counter-intuitive: when a user is really far away from both entities ($d_1 \gg y$), we expect them to visit the two entities with almost equal probability. This property makes an exponential model (or, for that matter, any superpolynomial model) undesirable.

### 4.2 Language models

In this work we do not attempt to model the differences in the language usage of different users. That is, we assume that the language model is independent of the user. Formally, given the entity, the probability of observing a given text is independent of the user:

$$P(t \mid e, u) = P(t \mid e).$$  \hspace{1cm} (5)

This is a reasonable assumption — otherwise we will run into data sparsity issues since most of the users will not have sufficiently many tweets to learn user-specific language models. Under this assumption, Equation (1) is reduced to

$$P(e, t, u) = P(t \mid e, u)P(e, u) = P(t \mid e)P(e, u).$$  \hspace{1cm} (6)

Given an entity $e$ and a tweet $t$, $P(t \mid e)$ denotes the probability of observing the words in the textual content of the tweet, given that the tweet is about entity $e$. For example, if $E$ is the set of restaurants and $e$ is a specific restaurant, then the tweet is likely to include words from the attributes such as name, address, and cuisine of $e$, as well as generic words such as dinner and delicious that are associated with restaurants.

Several unigram-based language models have been considered in the past for entity-matching, including simple term-frequency based models [5] as well as more advanced translation models [6]. We consider a model that is an extension of the language model proposed by Dalvi et al. [5], which we refer to as the unigram mixture model. We briefly describe the model here.

### 4.2.1 Unigram mixture model

Given a set of known entities from a specific category, e.g., a set of restaurants, the unigram mixture model represents the probability distribution of documents (in our case, tweets) that are about these entities. Here, both the document and the entities are modeled as a bag of words. Given a document $t$ about an entity $e$, each word in $t$ is drawn randomly from a mixture of two language models: a generic unigram language model $P_{lm}$ for the domain which is independent of $e$ (e.g., words related to restaurants) and an entity-specific language model $P_e$ which is a distribution over words in $e$ (e.g., the name and address of $e$). Given a mixture coefficient $\theta \in (0, 1)$, it assigns

$$P(t \mid e) = \prod_{w \in t} (\theta \cdot P_e(w) + (1 - \theta) \cdot P_{lm}(w)).$$

$P_{lm}$ can be learnt by counting the word frequencies of documents in the collection, while $P_e$ can simply be taken as the uniform distribution over the set of words in $e$.\textsuperscript{9}

\textsuperscript{9}If we have different attributes in $e$, we can give different weights to words in different attributes in $P_e$, e.g., words in names can have more weight than the words in addresses.
case of full-length documents, the redundancy in the document compensates for this information loss in representation. However, in our setting, since the tweets are very short and most of the times entity names are mentioned just once, we need to make use of the context of name mentions to improve accuracy. Thus, we consider an extension which we will refer to as the bigram mixture model.

4.2.2 Bigram mixture model

Bigram language models (more generally, n-gram language models) are well-studied, but combining them into a generative mixture model for entities is tricky. Standard bigram models (more generally, n-gram language models) are well-studied, but combining them into a generative mixture model for entities is tricky. Standard bigram language models represent the probability of a sequence of words $w_1, \ldots, w_n$ as

$$P(w_1, \ldots, w_n) = \prod_i P(w_i | w_{i-1}).$$

Let $f(w_i | w_{i-1})$ be the frequentist estimation, which is the number of occurrences of the sequence $(w_{i-1}, w_i)$ divided by the number of occurrences of $w_{i-1}$ in the data. Usually this is smoothed with a unigram term due to data sparsity,

$$P(w_i | w_{i-1}) = \delta \cdot f(w_i | w_{i-1}) + (1 - \delta) \cdot f(w_i).$$

Let $P_{lm}$ and $P_e$ be the bigram language models for the generic language and entity $e$. We need to adapt the standard bigram language models to account for the fact that $w_i$ and $w_{i-1}$ can be drawn from either $P_{lm}$ or $P_e$. For each $w_i$, there are four possibilities: it is a continuation of a bigram from $P_{lm}$, it is a continuation of a bigram from $P_e$, it is the start of a new bigram from $P_{lm}$ while the previous word was from the entity $e$, or it is the start of a new bigram from $P_e$ while the previous word was from the generic language.

To take this into account in defining the bigram mixture model, we introduce a new token called ENTITY, which stands for the entity words. In $P_{lm}$, in addition to the usual bigram sequences $(w, w')$, we also have bigrams of the form $(w, ENTITY)$ and (ENTITY, $w$). Thus, in our model, in addition to better modeling the sequential information in language usage, we model to an extent the context for an entity mention. For instance, for restaurants, we expect a high probability for bigrams such as (at, entity), (in, entity), and (ENTITY, has).

Let $q_e(w)$ denote the posterior of $w$ being drawn from $P_e$.

$$P(t | e) = \prod_i \left[ q_e(w_{i-1}) \cdot A + (1 - q_e(w_{i-1})) \cdot B \right],$$

where

$$A = \theta \cdot P_e(w_i | w_{i-1}) + (1 - \theta) \cdot P_{lm}(w_i | ENTITY),$$

$$B = \theta \cdot P_{lm}(ENTITY | w_{i-1}) + (1 - \theta) \cdot P_{lm}(w_i | w_{i-1}),$$

$$q_e(w_i) = q_e(w_{i-1}) \cdot \theta \cdot P_e(w_i | w_{i-1}) + (1 - q_e(w_{i-1})) \cdot \theta \cdot P_{lm}(ENTITY | w_{i-1}).$$

4.3 The estimation and inference problems

Let $\Delta$ denote the parameters of the distance model that include the set $\{a(u) | u \in U\}$, the set $\{b(e) | e \in E\}$, $d_0$, and $k$ from Equation (2). Let $\Delta$ denote the parameters of the language model that include $P_{lm}$, $P_e$, and $\theta$. Note that $\Delta$ and $\Lambda$ completely specify the probability model.

We assume that we know the location $l(e)$ of each entity $e \in E$, which can be obtained by geomapping its physical address. However, the locations $l(\cdot)$ of users are unknown; let $L$ denote the mapping of all users to locations. Let $T$ be a set of tweets consisting of $(t, u)$ pairs (note that we do not have any (entity) labels for the tweets). We consider the following estimation problem.

**Problem 1.** (Estimation) Given $T$, determine parameters $\Delta$, $\Lambda$, and the locations $L$ of all users that maximize the likelihood of observing $T$.

Once we learn the model, we want to use it to match tweets with entities. Given a tweet $t$ and a user $u$, we want to find the most likely restaurant (if at all) the tweet is about. This leads to the following inference problem.

**Problem 2.** (Inference) Given a tweet $t$ and a user $u$, compute

$$\arg \max_{e \in E} P(e | t, u),$$

i.e., the most likely entity given $t$ and $u$.

5. MODEL ESTIMATION

Given a set $T$ of tweet–user pairs, we want to estimate parameters $\Delta$, $\Lambda$, and the locations $L$ of all users that maximize the likelihood of observing $T$, i.e., find

$$\arg \max_{\Delta, \Lambda, L} \prod_{(t, u) \in T} P(t, u).$$

We have $P(t, u) = \sum_{e \in E} P(e, t, u)$, where $P(e, t, u)$ is given by Equation (6). Thus, we want

$$\arg \max_{\Delta, \Lambda, L} \prod_{(t, u) \in T} \sum_{e} P(t | e)P(e, u).$$

The main difficulty in solving Equation (9) lies in the normalization term in Equation (4). Since the term depends on the specific location of each user and each entity in relation to each other, it is hard to compute it exactly. We therefore introduce a set of simplifying assumptions that lead to reasonable interpretations of the model in order to make the problem tractable.

5.1 Making the estimation tractable

We make the following smoothness assumptions to make the model tractable while remaining interesting and useful.

1. We assume that the density of entities in any small neighborhood is proportional to the density of users in that neighborhood.
2. Given any region, the $\alpha$ values of the users come from a distribution $\Gamma_\alpha$, which is independent of the region.
3. Given any region, the $\beta$ values of the entities come from a distribution $\Gamma_\beta$, which is independent of the region.

The first assumption is intuitive, as densely populated regions have more businesses. The second assumption states that each region has a similar mix of users in terms of their interests in tweeting about restaurants. Likewise, the third assumption states that each user has access to a similar mix of popular and unpopular entities. These are intuitively true
for any given region. Now, we describe how these assumptions lead to a natural interpretation for the α and β parameters, and how they can be estimated efficiently.

Recall the definition of \( g(e, u) \) given in Equation (3). For a user \( u \), define

\[
F(u) = \sum_e \beta(e) (d_0 + d(e, u))^{-k}.
\]

Similarly, for an entity \( e \), define

\[
G(e) = \sum_u \alpha(u) (d_0 + d(e, u))^{-k}.
\]

Also, while the entities and users are discrete points in space, we approximate their distributions \( \Gamma_u \) and \( \Gamma_e \) by assuming them as continuous functions. Let \( \gamma_u \) and \( \gamma_e \) be the means of the distributions \( \Gamma_u \) and \( \Gamma_e \) respectively. For simplicity, the rest of this section assumes that the densities of the entities and the users are constant, and given by \( \tau_e \) and \( \tau_u \) respectively. The analysis can be extended to the case when the ratio of the user density to the restaurant density is a global constant.

First, we show that under our assumptions, \( F(u) \) is a constant independent of \( u \).

**Lemma 3.**

\[
F(u) = \frac{2\pi \tau_e \gamma_e}{(k - 1)(k - 2)d_0^{k - 2}}.
\]

**Proof.** Since we assume a continuous distribution of entities in the plane, to compute \( F(u) \), we integrate it over the plane. Consider a small disc centered at \( t(u) \) of radius \( x \) and infinitesimal width \( dx \). The area of the disc is \( 2\pi x dx \). Total number of entities in the disc equals \( \tau_e \cdot 2\pi x dx \). The average value of \( \beta(e) \) in this region is \( \gamma_e \). Thus, we have

\[
F(u) = \int_0^\infty \gamma_e (d_0 + x)^{-k} \cdot \tau_e \cdot 2\pi x dx
\]

Solving the above integral gives us the expression in the lemma. \(\square\)

Likewise, we show that under our assumptions, \( G(e) \) is a constant independent of \( e \).

**Lemma 4.**

\[
G(e) = \frac{2\pi \tau_u \gamma_u}{(k - 1)(k - 2)d_0^{k - 2}}.
\]

**Proof.** Similar to the above. \(\square\)

The main conclusion of the above two lemmas is that the terms \( F(u) \) and \( G(e) \) can be assumed to be constant over the set of users and set of entities respectively. This allows us to interpret \( \alpha(u) \) and \( \beta(e) \) naturally. For example,

\[
P(e) = \sum_u \alpha(u) \beta(e) (d_0 + d(e, u))^{-k} = \beta(e) G(e) \alpha(u) \beta(e),
\]

since \( G(e) \) is independent of \( e \) by Lemma 4. Also, the parameters \( \alpha \) and \( \beta \) are scale invariant model parameters, i.e., if we scale each \( \alpha \) by some constant and, independently, each \( \beta \) by another constant, we get back the same model. Thus, we can take \( \beta(e) \) to be exactly equal to \( P(e) \), i.e., the probability that a random tweet is about the entity \( e \). Similarly, \( \alpha(u) \) can be taken as simply the probability that a random tweet is from the user \( u \).

### 5.2 Learning the parameters

In this section we describe how to solve Equation (9). We can write the objective function as

\[
\prod_{(t, u) \in T} \sum_e P(t \mid e)P_e(u) = \prod_u P(u) \prod_{(t, u) \in T} \sum_e P(t \mid e)P_e(u).
\]

Note that \( P(u) = \alpha(u) \) as we approximated in the previous section. Similarly, \( P_e(u) = \beta(u) (d_0 + d(e, u))^{-k} / G(e) \), which does not depend on the \( \alpha \) parameters. Thus, we can independently optimize the first product and the rest of the expression, since they do not share parameters. Since \( \alpha(u) = P(u) \), we can estimate it as the fraction of all tweets that are by the user \( u \), which can be directly computed from the set \( T \) of tweets. For the rest of the section we focus on the following optimization:

\[
\arg \max_{\Delta, \Lambda, L} \prod_{(t, u) \in T} \sum_e P(t \mid e)P_e(u).
\]

Since the optimization function is in a product of sum form which is difficult to optimize directly, we apply the expectation-maximization (EM) method to simplify its form. We define a hidden random variable \( \pi(t, u, e) \) for each triplet:

\[
\pi(t, u, e) = \begin{cases} 1 & \text{if } (t, u) \text{ is about } e, \\ 0 & \text{otherwise}. \end{cases}
\]

Then, we can write our objective function as

\[
\arg \max_{\Delta, \Lambda, L} \prod_{(t, u) \in T} \sum_e (P(t \mid e)P_e(u))^{\pi(t, u, e)}.
\]

Now we can apply EM iteratively.

**E step.** In the E step, we assume some initial value of the parameters, say \( \Delta_0, \Lambda_0, L_0 \) and find the expected value of the hidden variables. We have,

\[
E(\pi(t, u, e)) = P_e(u) = \frac{P_e(t, u)}{\sum_{e'} P_e(t, u')}.
\]

Using the current values of the parameters, we can compute the above expression. In theory, this can be done by iterating over each entity for each tweet \( (t, u) \). This, however, will be infeasible in practice. Instead, for each tweet, we maintain a set of candidate entities that have a non-trivial overlap with the text of the tweet, and assume the probability of match to be 0 for all other entities.

**M step.** In the M step, we use the values \( f(t, u, e) = E(\pi(t, u, e)) \) and find the parameters \( \Delta, \Lambda, L \) that maximize the following function:

\[
\prod_{(t, u) \in T} (P(t \mid e)P_e(u))^{f(t, u, e)}
\]

\[
= \left( \prod_{(t, u) \in T} P(t \mid e)^{f(t, u, e)} \right) \left( \prod_{(t, u) \in T} P_e(u)^{f(t, u, e)} \right).
\]

We see that the first product only involves the language parameters \( \Lambda \), while the second product only involves \( \Delta \) and \( L \), the parameters of the distance model. Thus, we can separately optimize the language model and the distance model.
Learning the distance model. We learn \( d_0 \) and \( k \) by directly fitting the observed data, as described in Section 7. For user locations, we take all possible locations of the tweets (i.e., locations of all possible candidate entity matches for the tweet) from the user and compute the second product in Equation (10) from each of the locations to find the optimal location of the user.

Learning the language model. We can learn the bigram language model by a simple frequency counting. We keep a count of all the bigrams. Consider a \((t, e)\) pair, with the value \( f(t, u, e) \) as computed in the E step. First, we look for the name of entity \( e \) in the tweet, and replace it by a single token called ENTITY. If there is no exact match, we look at the maximal common substring of tokens between the tweet and the entity, and treat that as the reference to the entity name. After the replacement, for each bigram in the tweet, we add an amount of \( f(t, u, e) \) to the global count of that bigram. We process all the \((t, e)\) pairs, and finally, normalize all the bigram counts to sum up to 1. The resulting distribution is the language model \( \mathbf{P}_{lm} \) that maximizes the first product term in the M step.

6. DATA

In this section we first describe the data, which consists of tens of billions of tweets recorded over a year. The set of entities we consider are the restaurants in the continental United States.

We use the Twitter data, which consists of all the tweets with the associated metadata such as the name of the user, location of the user (if available), time, etc. We considered the data from December 2009 to January 2011. We consider the set of restaurants in the US, where each restaurant is associated with metadata such as the name, address, phone number, cuisine, etc. This data is obtained from Yahoo! Local (local.yahoo.com) and consists of about 750K restaurants.

To illustrate the efficacy of our methods, we focus on tweets that pertain to restaurants. Given the large volume of tweets, only a small portion of them will be about restaurants. Ideally we would like to focus only on tweets that have a non-negligible chance of being about restaurants. Developing a highly accurate classifier for this purpose is not easy. Previous work has exploited the hashtags in tweets to generate labels for certain classification tasks [7]; for example, tweets with the hashtag \#hcr can largely be assumed to be related to the health-care reform. Most of these hashtags have evolved by convention and usage. Unfortunately, we are not aware of a set of good hashtags that can help in our restaurant classification task. In fact, the top hashtags for the restaurant-related tweets (as determined by the filtering method described below) are \#Yelp, \#coupon, \#food, \#restaurant, \#coupon, and they cover less than 15% of the tweets. Since the main focus in this paper is to investigate the effectiveness of our model, we consider developing such classifiers as an orthogonal (but interesting) effort to solve the final task, and opted for a simple keyword-based filtering scheme as described below.

We obtained the textual content of about 300K Yelp restaurant reviews used in [5], and picked the top occurring words in the reviews (discarding the stopwords) as our keyword list. The keywords are: \{cafe, restaurant, club, bar, lounge, dinner, lunch, food, pizza, grill, coffee, sushi, tavern, kitchen, breakfast, diner, yelp, taco, burger, drinks, drink, brunch, bbq, bakery, yummy, burrito, cheese, tasty, delicious, crowded, eat, eating\}. Tweets that do not contain any of these keywords were filtered out from our collection. As we will discuss in Section 8, even this seemingly strict filtering scheme results in many tweets that are not about restaurants.

We collected all tweets with a non-trivial match to a restaurant name from our set of restaurants (i.e., matching words more uncommon than food, which is the name of three restaurants in our set of restaurants). This yielded a dataset of over 200 million tweets, authored by close to 14 million users.

7. EXPERIMENTS: DISTANCE MODEL

In this section we examine the distance model with which we infer the geographic locations of users. We perform two experiments. The first examines how well our polynomial distance model fits the data. The second examines how accurate the inferred user locations are.

We first look at how our distance model fits the observed data in terms of the distribution of user–entity distance. More specifically, suppose we have an oracle tweet–location mapping (e.g., using tweets that are geotagged), does the polynomial decay function in the distance model accurately explains the observed data? We consider the subsets of tweets that are geotagged; around 3% of all tweets fall into this category. For each unique user who authored one of these tweets, we compute the user location by optimizing the EM objective function given by our distance model. We then compute the distance of each tweet to the inferred location of its author. Figure 2 shows the resulting distribution. We also plot the distribution as predicted by the model, i.e., \((d_0 + x)^{-k} \cdot x\): the probability that a user tweets about a restaurant at a distance \(x\) is proportional to \((d_0 + x)^{-k}\), and the number of entities at distance \(x\) is proportional to \(x\), assuming a uniform distribution of entities. We compute the parameters \(d_0\) and \(k\) that best fit the data. Figure 2 plots both the observed and the theoretical distributions, and we see that predicted distribution fits the observed distribution closely; here, \(d_0 = 6\) and \(k = 3.2\).

![Figure 2: Observed probability distribution vs. distance model.](image)

In the second experiment, we measure the performance of our distance model in inferring the actual user locations.
To this end, we focus on users who have issued at least one restaurant-related tweet (of a restaurant in our collection) and have self-reported their geolocations by explicitly specifying a latitude-longitude pair. Since each qualifying tweet carries a user location in the metadata, we obtain a distribution over locations for each such user; let \( \ell(u, \cdot) \) denote this distribution. This serves as the ground-truth. We then apply the distance model to infer the location of each user in this collection, withholding the ground-truth location from the model. We measure the error of the inferred location \( \ell'(u) \) against the ground-truth in two ways. In the min measure, the error is the minimum distance between \( \ell'(u) \) and \( p \), where \( p \) is in the support of \( \ell(u, \cdot) \), i.e., \( \ell(u, p) > 0 \). In the mode measure, it is the distance between \( \ell'(u) \) and \( p \) where \( p \) is the mode of the distribution \( \ell(u, \cdot) \).

Figure 3 shows the performance curves as a cumulative fraction: the x-axis is the distance in miles and the y-axis is the fraction of users whose error is within the distance. We study the error for users with at least 3 and at least 10 tweets for a peripatetic user with a self-reported location in the collection. From disparate locations, the inferred user location is within a mile of the self-reported location.

Figure 4: Location of the restaurant tweets for a user from Atlanta, GA.

These experiments illustrate that our distance model not only accurately captures the true distance distribution of tweet-user pairs but also is useful in obtaining the actual location of the user fairly accurately.

8. EXPERIMENTS: MATCHING

8.1 Experimental setup

In this section we evaluate the performance of the model for the matching task. Note that our model outputs the entity \( e \) that maximizes the probability of generating a given tweet. However, for some tweets, we want to return a “no match”, if the most likely entity results in a low probability. To this end, we add a dummy entity in our collection, Non-Restaurant, which has no distance model but has a language model consisting of the entity-independent language \( P_m \). In addition, we assign a prior probability \( c \) to this object. The higher the value of \( c \), the more likely a random tweet is non-restaurant. If during matching, the most likely entity is found to be Non-Restaurant, we say that the tweet has “no match”. As default, we set \( \theta = 0.2, c = 0.999 \).

Test data. If we randomly sample from the tweet collection that has been filtered by the keyword list as described in Section 6, then most are either non-restaurant or do not contain a match to any of the candidate restaurants in our dataset. On the other hand, if we take only the tweet–match pairs with the highest confidence, then we do not get to evaluate our performance on the low-confidence end. In order to obtain a reasonable amount of positive data points with limited human labeling effort, we need to select across the board. To this end, we took the raw score produced by the model in [5], ranked the tweet–match pairs according to this raw matching score\(^7\), and selected the same amount of examples to label from each of the following buckets: top 2%, 2–5%, 5–30%, and the rest. We filtered out users with more than 10K tweets as these users are less likely to be real users. 490 tweets sampled this way were labeled in the following procedure. Each tweet, along with all candidate restaurants, is presented to annotators, who either label the tweet as not about any of the candidates (e.g., not a restaurant tweet), or choose one of the candidates as the correct match. The location information of the user, if available, is also presented. In order to give more contextual information

\(^7\)Note that the original model was developed to select the best matching restaurant for documents that are known to be restaurant reviews, thus the raw score is not optimal for this purpose. Nonetheless, it is a good surrogate to obtain a reasonable sample from the dataset.
about the user, we also provide other tweets written by the user in the past.

We compare different methods by comparing the precision–recall curves. We compute the precision–recall curve in the following manner. For each tweet $t$, we take the candidate restaurant $r$ with the highest matching probability $p$ and form a $(t, r, p)$ tuple. For a given threshold $\tau$, we output $r$ for $t$ if $p > \tau$ and for all tweets with $p \leq \tau$, output “no match”. If $r$ is the match chosen for $t$ by annotators, we consider this to be a correct match. Note that this slightly differs from a standard precision–recall curve in that the recall is.

8.2 Results

First, we compare the performance of the language model with and without the distance model. As we can see in Figure 5, with our default parameter setting ($\theta = 0.2, c = 0.999$), we observe clear improvements in the precision–recall curve by utilizing the distance model. This trend is consistent for other $c$ values as well (Figures 6(a) and 6(b)).

In addition, we investigate the effect of using different parameters. This set of comparison is done without using the distance model. First, we fix $\theta = 0.2$ and experiment with different values of $c$. As shown in Figure 7, we achieve better performance with large values of $c$. If we hope to maintain at least 50% in recall, then the optimal value for $c$ ranges between $1 - 10^{-7}$ and $1 - 10^{-10}$. As we increase $c$ to $1 - 10^{-10}$, we observe a clear degradation in performance on the higher recall range. Indeed, for $c$ ranging from $1 - 10^{-6}$ to $1 - 10^{-10}$, we observe higher $c$ values performing better at the lower end of recall and lower $c$ values performing better at the higher end of recall. This is a reasonable behavior since higher $c$ values are in effect favoring tweets with extremely high confidence in the matching, and user locations are in turn computed with more accurate matches.

We then compare different values of $\theta$ for $c = 1 - 10^{-9}$. Figure 8 shows that for a wide range of $\theta$ values (0.2–0.4), the performance is quite stable.

Finally, We observe that the performance degrade if we switch to the unigram model. Using the default parameter setting, we see that the bigram model outperforms the unigram model when used on its own (Figure 9(a)) as well as when used in conjunction with the distance model (Figure 9(b)). While the distance model improves both the unigram and the bigram models, the gap between them is larger in Figure 9(b). This trend is more pronounced for a higher value of $c$; we omit these plots due to space constraints.

9. CONCLUSIONS

In this paper we formulated a user model for generating a tweet about an object. Our model depends on three intuitive quantities: the propensity of a user to tweet about an object, the object’s inherent popularity, and the distance between the user and object’s geographic location. We obtained an efficient estimation algorithm that can infer the model parameters. We then used this algorithm to match objects to tweets, focusing on the restaurant domain. Our experiments indicate that the model and methodology we developed work well in practice. As a byproduct of our modeling, we also obtained a principled method to infer the location of a user using the restaurant tweets; this inference method may be of independent interest.

This object matching primitive is essential for applications that associate an object with tweets. These applications promise to be useful to the web community at large. Future work includes fine-tuning the model to take into account multiple locations for a user (home vs office, travel, etc) and including a user-specific popularity for a restaurant (preference for a cuisine, decor, etc).
10. REFERENCES


