Like Prediction: Modeling Like Counts by Bridging Facebook Pages with Linked Data

Shohei Ohsawa
University of Tokyo
2-11-16 Yayoi, Bunkyo-ku
Tokyo, Japan
ohsawa@weblab.t.u-tokyo.ac.jp

Yutaka Matsuo
University of Tokyo
2-11-16 Yayoi, Bunkyo-ku
Tokyo, Japan
matsuo@weblab.t.u-tokyo.ac.jp

ABSTRACT

Recent growth of social media has produced a new market for branding of people and businesses. Facebook provides Facebook *Pages* (Pages in short) for public figures and businesses (we call entities) to communicate with their fans through a *Like* button. Because Like counts sometimes reflect the popularity of entities, techniques to increase the Like count can be a matter of interest, and might be known as social media marketing. From an academic perspective, Like counts of Pages depend not only on the popularity of the entity, but also on the popularity of semantically related entities. For example, Lady Gaga's Page has many Likes; her song "Poker Face" does too. We can infer that her next song will acquire many Likes immediately. Important questions are these: How does the Like count of Lady Gaga affect the Like count of her song? Alternatively, how does the Like count of her song constitute some fraction of the Like count of Lady Gaga herself?

As described in this paper, we strive to reveal the mutual influences of Like counts among semantically related entities. To measure the influence of related entities, we propose a problem called the *Like prediction problem* (LPP). It models Like counts of a given entity using information of related entities. The semantic relations among entities, expressed as RDF predicates, are obtained by linking each Page with the most similar DBpedia entity. Using the model learned by support vector regression (SVR) on LPP, we can estimate the Like count of a new entity e.g., Lady Gaga's new song. More importantly, we can analyze which RDF predicates are important to infer Like counts, providing a *mutual influence network* among entities. Our study comprises three parts: (1) crawling the Pages and their Like counts, (2) linking Pages to DBpedia, and (3) constructing features to solve the LPP. Our study, based on 20 million Pages with 30 billion Likes, is the largest-scale study of Facebook Likes ever reported. This research constitutes a new attempt to integrate unstructured emotional data such as Likes, with Linked data, and to provide new insights for branding with social media.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database applications-Data mining; J.4 [**Computer Application**]: Social and Behavioral Sciences

Keywords

Entity Linking, Facebook, Feature Construction, Link-based Prediction, Linked Data

WWW 2013 Companion, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2038-2/13/05.

1. INTRODUCTION

Recent growth of social media has produced a new market for personal branding. In this personal branding movement, people are coded into brands complete with promises of performance, specialized designs, and tag lines for success [7]. Several public figures such as book authors, musicians, and politicians devote attention to social media for personal branding use. In the recent US presidential campaign, President Barack Obama used social media to accumulate supporters. His official Twitter account and Facebook account, which has 23 million followers, publishes his activities with respect to national politics. Especially, Facebook provides dedicated accounts, Facebook *Pages* (Pages in short), for people and businesses to communicate and maintain relationships with their fans. By clicking a *Like* button on the Page, the fan can subscribe to its timeline and notify a friend that he supports it. A famous US singer, Lady Gaga, has her own Page, which is "Liked" by as many as 50 million users.

Because Like counts sometimes reflect popularity of the person or the business and because widely subscribed accounts can gain strong influence, methods to increase the Like count can be a matter of interest. Many social media companies commercially provide solutions to increase Likes, widely known as social media marketing. They sometimes give advice to make frequent updates, conduct campaigns, and give rewards to users. Several studies have examined the motivations of users in clicking like buttons. Those results are applied to effective web promotion [5].

From an academic perspective, it is interesting to ascertain the whole picture of Like counts. What category of Pages will acquire more Likes? What does the Like distribution look like? More technically, based on the recent advances of Semantic Web, how does the semantic relation of Pages influence Like counts? Because the Pages are mutually related semantically, the Like count of a Page depends not only on the popularity of the person, but also on the popularity of semantically related persons, organization, products, and so on (entities). For example, Lady Gaga's Page has many Likes; her song "Poker Face" also has many Likes. They are semantically related. It is natural to infer that the Like count of Lady Gaga affects the Like count of "Poker Face" and vice versa. The popularity of Lady Gaga implies that her new song will acquire many Likes immediately.

The key questions we address in this study are the following: How does the Like count of Lady Gaga affect the Like count of her songs? Alternatively, how does the Like count of her songs constitute the Like count of Lady Gaga herself? These mutual influences of Likes among entities (or the brand power of entities from the marketing perspective) are ubiquitous in many domains: For instance, Steve Jobs became popular largely because of his company's products such as the Macintosh computer, iPhone, and iPad. Simultaneously, Apple Computer Inc. products are attractive (at least partly) because of Steve Jobs. WWW2013 is famous as a

prestigious academic conference in computer science because past WWW conferences produced many good studies. In addition, the past WWW conferences are highlighted more because the succeeding conferences become increasingly popular.

As described in this paper, we strive to reveal the mutual influences of Like counts among semantically related entities. To measure the influence of related entities, we propose a problem called the *Like prediction problem* (LPP). It models the Like count of a given entity using information of related entities. The semantic relations among Pages are not described in Facebook. Consequently, our approach is to obtain mapping of each Page with the most similar DBpedia article. DBpedia provides relations among entities as RDF [13] predicates. Therefore, we can infer a relation among Pages. We apply support vector regression (SVR) to LPP and obtain the learned model. Using the model, we can estimate the Like count of a new entity e.g., Lady Gaga's new song. More importantly, we can analyze the model and find which RDF predicates are important to infer the Like count. The important predicates constitute a *mutual influence network* among entities, which provides a comprehensive view of Like count influences.

Our study has three parts: (1) crawling the Pages and their Like count, (2) linking Pages to DBpedia, and (3) constructing features to solve an LPP. To crawl the data, we design a focused crawler. It collected 20 million Pages and 30 billion Likes, covering at least 25.8% of all Pages and 60.5% of all Likes of pages. For linking Pages to DBpedia, we propose a method for entity linking to find good correspondence between Pages and DBPedia. The 8.8 million concepts in DBpedia are connected to Likes, with greater than 91% accuracy. To produce a regression problem, features are constructed using information of related entities, such as the total number of Likes of the entities with a certain predicate. Like counts can be predicted with 8.1% root mean squared error.

This study is the largest-scale examination of Facebook Likes ever reported. Although confined to Facebook, this research represents a new attempt to integrate unstructured emotional data, e.g., Likes, with Linked data. Moreover, it provides new insights for branding with social media.

Section 2 presents a description of an overview of Pages and the design rationale of our focused crawler. Section 3 defines LPP and a method for the problem and explains the architecture of entity linking and feature construction, with details in Section 4. Section 5 presents experimentally obtained results. After describing related work and discussion, we conclude the paper.

2. OBTAINING FACEBOOK PAGES

To analyze Like counts of Pages, we must obtain data. We take an approach of using publicly available Facebook APIs. To follow the rules and the limitations of API, we must make many devices. In this section, we explain the crawler architecture. Although this is an engineering work and does not convey much academic contribution, knowledge about our crawler is important to understand the data in the following sections. Furthermore, our method can be useful to obtain social media data using API following the restriction and limitation for academic purpose. After describing the crawler, we present basic statistics related to Pages.

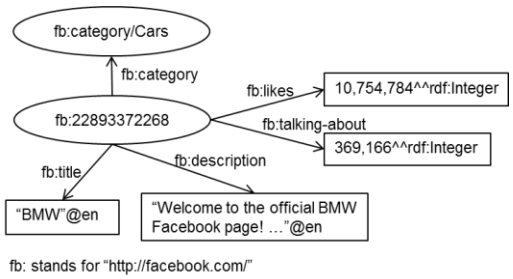


Figure 1: Structure of Facebook RDF.

2.1 Focused Crawler

First, we explain Facebook API. We can obtain data on a Page if we make a query using the API (we use *Graph API*[1]). Fig. 1 depicts the data structure of Page represented using RDF. The properties of pages are ID, title, category, description, a Like count, and a talking-about count. The title describes a theme of the page. No unique constraint exists for the title column. Therefore, two or more Pages might share the same title. Category is represented by texts. Description is the text about the Page. The most important information for our research, Like count, is the number of users who clicked a Like button on the Page. Talking-about count is the number of users mentioning the page on their walls, which show preferences, during the prior seven days.

The crawling approach we take in this study is using the search function of Graph API. This function returns pages having a title including query strings ordered by their relevance to a query. Varying the query words, we can obtain various pages corresponding to query words. A search function returns 5.4 pages on average. We designate this approach *dictionary-based crawling*, because, if we prepare a dictionary, we can fetch data by inputting each term in the dictionary, which yields a list of Pages.

2.2 Overview of Pages

As a dictionary, we used all English DBpedia entities. In all, 9,816,747 generated queries were used. We crawled 22,616,574 Pages in 190 categories. These Pages have 30,685,646,909 Likes in all. To crawl the data, we set up 20 Amazon EC2 instances and started crawling from 16 June 2012 23:00. All crawling tasks were completed after 9 days and 19 hours.

Table 1(a) shows the top 10 Pages. Eight musicians are in the top 10, including Rihanna and Lady Gaga. The Page with most Likes is Facebook's own Page, which seems natural because every user is a Facebook user.

Table 1(b) shows a Like count of each category. The Musician category has the most Likes. The second most Liked category is Community. Those pages in Community are for groups with similar preferences and goals to lists in Twitter. Facebook is often used by local businesses to maintain customer relationships, resulting on pages in Local business category. Facebook enables users to create an ad hoc page by entering the name of page in his profile page. For example, once a user types movies he likes in his profile page, Facebook automatically creates a page having the same title. Facebook categorize the page as Interest, which has the sixth most Likes.

[1] http://developers.facebook.com/docs/reference/api/

Table 1: Top 10 Liked Facebook entities

(a) Pages

Name	Category	Likes
Facebook	Product	68,600,026
Rihanna	Musician	57,657,090
Lady Gaga	Musician	52,079,088
Harry Potter	Movie	52,069,702
Shakira	Musician	52,009,693
Michael Jackson	Musician	50,165,651
Family Guy	Tv show	46,445,463
Katy Perry	Musician	44,334,697
AKON	Musician	40,409,740
Music	Field of study	40,372,722

(b) Categories

Category	Total Likes	Pages
Musician	3,389,510,858	898,115
Community	2,765,436,603	1,252,944
Local business	2,300,789,488	2,624,988
Movie	1,886,014,129	208,786
Public figure	1,728,130,793	1,339,680
TV show	1,426,823,633	141,145
Interest	1,206,517,853	2,623,121
Product	1,044,745,378	231,840
Company	934,409,058	789,186
Athlete	790,372,687	496,173

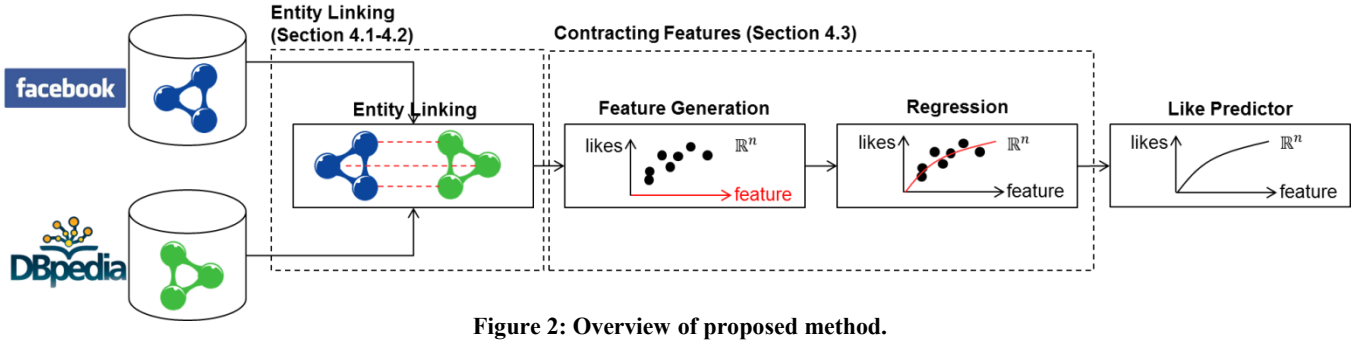


Figure 2: Overview of proposed method.

2.3 Performance of Crawlers

We use dictionary-based crawling. As a dictionary, we used all English DBpedia entities. In all, 9,816,747 generated queries were used. We crawled 22,616,574 Pages in 190 categories. These Pages have 30,685,646,909 Likes in all. To crawl the data, we set up 20 Amazon EC2 instances and started crawling from 16 June 2012 23:00. All crawling tasks were completed after 9 days and 19 hours.

Therefore, it is important to ascertain the performance of crawling. To ascertain the coverage, we randomly generate 10^7 IDs, test whether the corresponding pages exist. Results show that 466 Pages exist. Among the 466 Pages, we had already crawled 130 Pages. Therefore, the estimated coverage is 0.279±0.021, which shows that our crawling method covers at least 25.8% of all Pages with a 5% significance level. Similarly, the estimated coverage of Likes is 0.649±0.044.

The coverage is sufficiently large for analysis in the subsequent sections: First, the estimated coverage (0.646) is not a problem because we use the Like count as training data. The obtained model will not change much if the number of Likes is on average 0.646 times the real number. Second, the coverage of Pages (0.279) is also not a problem because it can be considered that the Like counts of related entities are visible in the 27.9% probability. Compared to the text categorization problem, where potentially related words do not necessarily appear often, 27.9% observation is not a bad result. Actually, the results of LPP do not change much if we crawl 100% of related entities.

3. LIKE PREDICTION PROBLEM

To ascertain the mutual influence of related entities, in this section, we define the Like Prediction Problem (LPP). We predict Like count of a given entity based on features of related entities. In this section, we give the definition of LPP, and the overall architecture of our method to solve LPP.

For example, A Lady Gaga's Page has 52,079,088 Like counts. Her song "Born This Way" obtained 100,043 Likes, "Poker Face" received 23,548 Likes, and "Bad Romance" received 117,916 Likes. Other than song entities, there are entities related to Lady Gaga such as her hometown, her parents, schools from which she graduated, and so on, each has its own Like count. For other singers such as Rihanna and Katy Perry, we can obtain the related entities and their respective Like counts. The LPP in this case is to estimate the Like of a singer given the Like counts of her songs, hometowns, schools from which she graduated, and so on. We can also use the property of related entities, such as the number of songs, the sales of songs, the number of students of graduated schools, and so on.

For singers, it is apparent that Like counts of the songs are important to estimate the Like counts of a singer, and the schools from which they graduate are less important. However, how about researchers? In this case, the graduate school might be more important as well as the Like count of a supervisor, or the current affiliation. The Like count of the song might not be influential to the Like count of himself as a researcher if a researcher happens to have written a song.

In LPP, we assume that Pages and related entities are all represented as resources in RDF graph. They have their own URIs.

The formal definition of Like prediction is the following.

Definition (Like prediction) For an labeled graph $G(V, E, L, a)$ where vertex set V, edge set E, possible edge labels L and a labeling map $a: E \to L$, Like prediction is defined as follows.

Input: A labeled graph $G(V, E, L, a)$, and possible category set $\mathcal{C}$ and training mapping $l_t: V_t \to R$

Output: An estimated regression function: $\hat{l}: V \to R$.

Like prediction can be boiled down to a problem of a *link-based regression with labeled edges* by coordinating RDF graph and Like counts.

Fig. 2 presents an overview of our method. After obtaining Pages and their count of Likes, we find matching between Pages and DBpedia URIs. Then, the related entities are obtained and

Table 2: Comparison of integration methods. Significance levels are 0.05 (*), 0.01 (**), and 0.001 (***). Bold indicates the best performance (p<0.05)

Method	Accuracy								
	Combined			uniform			homonymy		
	All	Selection	Rejection	All	Selection	Rejection	All	Selection	Rejection
Baseline	0.803	0.714	0.864	0.887	0.840	0.919	0.642	0.457	0.757
EL (Rank)	0.712	0.662	0.743	0.762	0.796	0.734	0.623	0.403	0.759
EL (Rank + Title)	0.882***	0.819***	0.925**	0.946	0.953***	0.943	0.757***	0.547**	0.888***
EL (Rank + Title + Description)	0.892***	0.847***	0.923**	0.952	**0.967*****	0.943	0.774***	0.602***	0.883***
EL (Rank + Title + Category)	0.906***	0.836***	**0.953*****	0.950*	0.953***	0.947	0.824***	0.605***	**0.961*****
EL-all (Rank + Title + Description + Category)	**0.916*****	**0.863*****	**0.951*****	**0.956****	**0.967*****	0.947	**0.841*****	**0.657*****	**0.956*****

features for LPP are constructed. After obtaining the features, we produce a regression model, and solve the problem by support vector regression (SVR). In the next section, we explain each process in detail.

4. METHOD

To make LPP, we must use related entities of a given Page as features. We first link a Page with corresponding DBpedia entity. Then we obtain relations of related entities. Then, the related entities are used to derive features for LPP. In this section, we explain entity linking and feature construction.

4.1 Entity Linking

It is not simple to find the appropriate DBpedia entity with a given Page. For example, we have 25 Pages entitled "Michael Jackson." This problem is known as homonymy in natural language processing. Therefore, we must find the link between entity and Page addressing this homonymy problem.

This task is known as entity linking. Several past studies cope with this problem by constructing probabilistic models [12]. Similar to past studies, we construct this model for social network analysis domain. We perform entity linking based on attributes of Page. Our model unifies several factors of Page such as a rank in search result, title, description and category. These factors are fed into one probabilistic model, which enables estimation on the most appropriate DBpedia entity for each Page on a maximum a posteriori (MAP) estimation manner. Our model considers the bipartite graph between all Pages F and all entities D and existing probability $p(e=1|f,d)$ of edge e based on similarity between $f \in F$ and $d \in D$. Our model selects $\hat{d}$ as an appropriate entity maximizing $p(e=1|f,\hat{d})$ for given f.

We model the $p(e=1|f,d)$ as a joint probability of the following probabilities.

Co-occurrence-based probability. This probability is based on co-occurrence of f and d in Facebook. We define rank r_{df} of a Page f for an entity d as defined as a position in search results by the query, just as in the title s_d, sorted by the count of Likes and the relevance to a given query. We define *search results* F_d of an entity d as a set of Pages that contains title s_d of d in its title s_f or description a_f. Formally, we define r_{df} as the following.

$$r_{df} = \begin{cases} 0 & (f \in F_d \text{ and } f \text{ is mostly Liked in } F_d) \\ r_{df'}+1 & (f \in F_d \text{ and } f \text{ is Liked next to } f') \\ \infty & (f \notin F_d) \end{cases}.$$

We model the distribution of r_{df} as the following exponential distribution: $PR(r_{df}|\lambda_1) = \lambda_1 \mathrm{e}^{-\lambda_1 r_{df}}$. Especially, if $f \notin F_d$, then f and d are not matched because $p(r_{df}) = 0$.

Text-based probability. We use a title s_f and description a_f of Page f. These features are text literals connected to subject f as an object by predicate `fb:title` and `fb:description`, respectively in Facebook RDF. Title s_d and description a_d of an entity d is connected by predicates `db:name` and `db:abstract` to d in DBPedia. We use an exponential distribution with regard to the text-based similarity and model the probability as

$$PT(\mathrm{sim}(s_f,s_d)|\lambda_2) = \lambda_2 \mathrm{e}^{-\lambda_2 \mathrm{sim}(s_f,s_d)}.$$
$$. PD(\mathrm{sim}(a_f,a_d)|\lambda_3) = \lambda_3 \mathrm{e}^{-\lambda_3 \mathrm{sim}(a_f,a_d)}.$$

As described in this paper, we use the Levenshtein distance as $\mathrm{sim}(\cdot,\cdot)$. Other similarities such as TF-IDF-based similarity are also compatible.

Category-based probability. We use a category $c_f \in C_F$ of a Page f The features are resources connected to a subject f as an object by predicate `fb:category`. Besides, we use a category $c_d \in C_D$ of an entity d. We define the joint probability of co-occurrence of two categories as $PC(c_f,c_d|\mathbf{M}) = M_{c_d c_f}$ where $\mathbf{M} \in \mathbb{R}^{|C_F|\times|C_D|}$ is a parameter matrix where its element is $M_{c_d c_f}$.

The main characteristic of our model is unifying not only the title but the rank, description, and category. The logarithm of $p(e=1|f,d)$ is decomposed as shown below.

$$\begin{aligned} &\log p(e=1|f,d) \\ &= \log PR(r_{fd}|\lambda_1) + \log PT(s_f,s_d|\lambda_2) + \log PD(a_f,a_d|\lambda_3) \\ &\quad + \log PC(c_f,c_d|\mathbf{M}) \\ &= -\lambda_1 r_{fd} - \lambda_2 \mathrm{sim}(s_f,s_d) - \lambda_3 \mathrm{sim}(a_f,\mathrm{a}_d) + \log M_{c_f c_d} + \mathrm{const.} \end{aligned}$$

Consequently, the logarithm of joint probability depends on a linear combination of features such as rank, title distance, description distance, and category likelihood.

To optimize the model parameter $\boldsymbol{\theta} = (\lambda_1, \lambda_2, \lambda_3, \mathbf{M})$, we conduct parameter estimation. We observed n combinations of variables $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$, where $\mathbf{y}_i = (r_{fdi}, s_{fi}, c_{fi}, a_{fi}, s_{di}, c_{di}, a_{di})$. Under MAP estimation, we estimated the optimal parameter.

We designate the *minimum confidence* $\hat{p}_r$. To estimate the optimal minimum confidence, we consider the 0–1 loss function as $L(\hat{r},\hat{p}_r;r) = \sum_{f\in F} I(\hat{r}(f,\hat{p}_r) \neq r(f))$. Here, $r: F \to \{0,1\}$ (F is

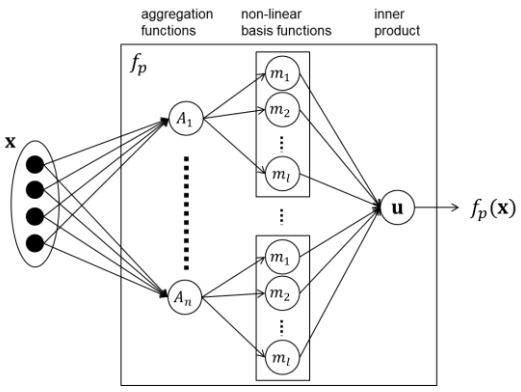


Figure 3: Like influence function.

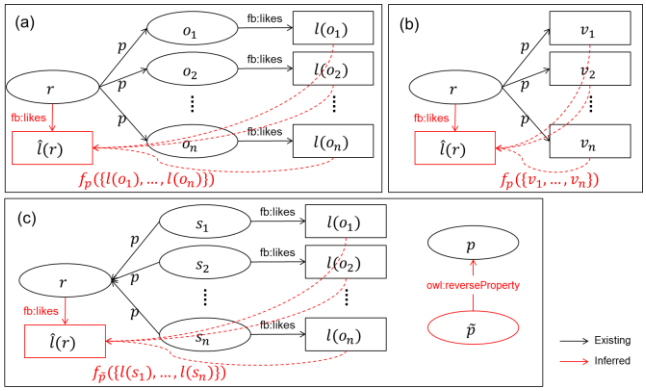


Figure 4: Effects of three types from related entities.

a set of all Pages) is the ground truth classifier that takes 1 and 0 if f does not select any entities and otherwise, respectively and $\hat{r}: F \times \mathbb{R} \rightarrow \{0,1\}$ is the learned classifier such that $\hat{r}(f, p_r)$ equals 1 if $p_r > \max_{d \in D} p(e = 1|f, d)$ and 0 otherwise. I is an indication function that takes 1 if a given proposition holds and 0 otherwise. To minimize the loss function presented above, we use the gradient descent method.

4.2 Evaluation of Entity Linking

The ground truth data are created manually. We developed an authoring system for the data and seven subjects used it. A subject solves the problem "what is the most appropriate DBPedia entities describing the proposed Page?" by looking at the Page on the top of the web page which is written the title, category and description and clicking a button on the right of corresponding DBPedia entities based on the title category and description (short abstract).

We asked questions to each of seven subjects. Each responded to 100–200 questions. The question set included questions of two types. The `uniform` questions were sampled randomly from all crawled Pages. Questions of the `homonymy` category include a Page with homonymous DBPedia articles. For example, "Chris Knight" is musician name and anthropologist name. Therefore, two entities exist: "Chris Knight (musician)" and "Chris Night (anthropologist)." We selected such cases having 15–40 homonymous entities randomly from our dataset. `uniform` and `homonymy` include 650 and 350 pages, respectively.

To evaluate the significance of our model, we compare the *accuracy* to those of other approaches. Here, the accuracy is a widely used criterion in entity linking tasks [12] defined for our purposes as correctly predicted pages among all pages.

We conduct an evaluation over all combinations of factors of rank, title, category, and description. Therefore, $2^4 - 1$ combinations can be examined including the baseline (the method using no factors is ignored). We evaluated the accuracy of model using ten-fold cross validation. Furthermore, to evaluate the accuracy of the reject classifier, we prove the true positive rate and false positive rate varying p_r. As the baseline, we employ an algorithm considering only a title distance. This algorithm calculates the Levenshtein distance between the page title and the title of each candidate named entity and selects the named entity having the least distance.

Table 2 presents a comparison of the aggregation methods. EL is described entity linking model. This table shows that EL-all is the most accurate in several combinations of features. Describing this table briefly, adopting the description feature to rank and title improves the accuracy by 1.0 points and adopting category features to three other features improves accuracy by 1.4 points. Furthermore, the proposed-all is more accurate than the Baseline by 11.3 points. This result suggests that our proposed approach is better than the Baseline approach.

4.3 Constructing Features

Once we obtain the linking of Page and DBpedia entity, we can obtain the related entities to a given Page. Then it is important how we can use the related entities to obtain features for LPP.

For a given URI r, we predict a target Like count $\hat{l}(r)$ the function of Like counts and literal values with regard to related entities. We model $\hat{l}(r)$ as $\hat{l}(r) = \sum_{p \in \mathbf{P}} f_p(V_{rp})$,

where **P** is a set of all predicates in given RDF. V_{rp} is a set of all value of objects (Like of resource and literal) with regard to s and p. We divide the features into three types as depicted in Fig. 4.

a) Like influence. This influence is considered for a triple (r, p, o) for predicate p and resource o. We define $V_{sp} = \{l(o_1), \ldots, l(o_n)\}$ for all resources that hold (r, p, o_i) in G_{RDF}.

b) Literal influence. This influence is considered for a triple (r, p, v) for predicate p and literal v. We define $V_{sp} = \{v_1, \ldots, v_n\}$ for all resources that hold (r, p, v_i) in G_{RDF}. We adjust the unit by multiplying constant if $v_1, \ldots, v_n$ have different data type each other (*e.g.* currency).

c) Reverse Like influence. This influence is considered for a triple (s, p, r) for predicate p and resource s. We infer $\tilde{p}$ as a reverse predicate of p. We define $V_{s\tilde{p}} = \{l(s_1), \ldots, l(s_n)\}$ for all resources that hold $(r, \tilde{p}, s_i)$ in G_{RDF}.

To obtain the features, we define a influence function as $f_p: 2^{\mathbb{R}} \rightarrow \mathbb{R}$ where $2^{\mathbb{R}}$ is a power set of $\mathbb{R}$ (*i.e.* a set of *all sets of* $\mathbb{R}$). We define $f_p(\mathbf{x})$ using following equation.

$$f_p(\mathbf{x}) = \sum_{i,j} u_{pij} m_i(A_j(\mathbf{x})) = \mathbf{u}_p^{\mathbf{T}} \boldsymbol{\phi}(\mathbf{x}),$$

where m_i is a basis function (e.g. linear, square root and log), A_j is an aggregation function (e.g. sum, average, count and max), $\mathbf{u}_p^{\mathbf{T}} = (u_{11}, \ldots, u_{1l}, \ldots, u_{n1}, \ldots, u_{nl})$ and $\boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} = (m_1 A_1(\mathbf{x}), \ldots, m_1 A_l(\mathbf{x}), \ldots, m_n A_1(\mathbf{x}), \ldots, m_n A_l(\mathbf{x}))$. Because this equation is slightly complicated, we show an interpretation of this equation in Fig. 3. The function consists of three stages. In the first stage, all members in input set are aggregated and modified in the second stage. In the final stage, each basis function are aggregated by inner product by $\mathbf{u}_p$.

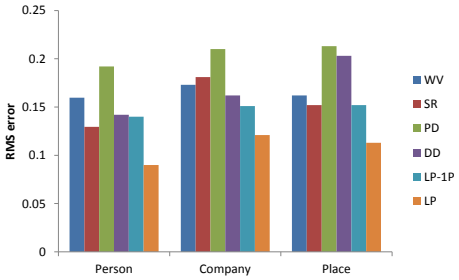


Figure 5: Performance of several prediction methods.

Careful examination reveals that this influence function depends only on $\mathbf{u}_p$. Therefore, we call this vector the *influence parameter* and define *influence* of p, $\|\mathbf{u}_p\|_{L1}$.

Especially, we define $f_{\text{owl:sameAs}}(\mathbf{x}) = \sum_{x \in \mathbf{x}} x$.

From the equations presented above, we can use the following equation.

$$\hat{l}(r) = \sum_{p \in \mathbf{P}} \mathbf{u}_p \boldsymbol{\phi}(V_{rp}),$$

By looking at the equation of Like prediction above, we can find that this problem is a linear regression problem.

Finally, we describe the training algorithm to learn the optimal parameter using given training data. As discussed in Section 3, the training data of Like prediction are a function of the correct Like count. To avoid overfitting, we use a support vector regression (SVR). We use a linear kernel because we already consider the basis function and estimate the optimal model parameter by cross validation.

5. EXPERIMENT

In this section, we first evaluate the LPP performance. Then, we investigate which predicates are important for LPP. Then, the mutually influential network (MIN) is shown.

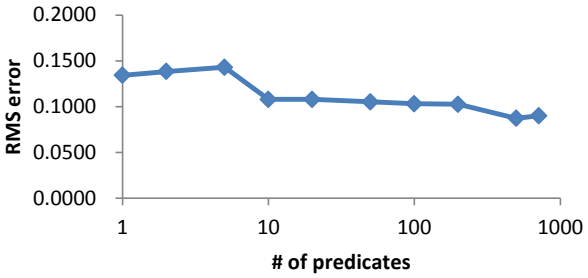
5.1 Performance of LPP

First, we perform evaluation for LPP. We use a dataset for the evaluation the integrated RDF dataset containing Pages and DBPedia. We perform four-fold cross validation and obtain the root mean squared (RMS) error in each validation as an evaluation criterion. We employ three datasets for evaluation: Person, Company and Place. Each dataset has 500,000 Pages and each corresponds to a DBPedia entity, which has the same type as the dataset name. To confirm the benefits of Like prediction, we compare our approach to each of the following two baselines.

Wikipedia viewed count (WV). This method estimates the Like count of a given Page by multiplying the viewed counts of Wikipedia articles of the same name. We adjust the multiplier using the least-squares approach.

Search result count (SR). This method estimates the Like count of a given Page by searching the title as a keyword and by obtaining a count of search results. We use Bing as the target search engine.

Page description (PD). This method uses a description of Pages as a feature. We employ a bag-of-words model and map the text into vector space using TF-IDF. Dimensions corresponding to the word and the value correspond to TF-IDF.



DBPedia description (DD). This method uses a short abstract of DBPedia as a feature. The vector space construction method is the same as that for PD.

Like prediction with a single predicate (LP-1P). This method is almost identical to Like prediction. It is only different in that it does not use a label of the predicate. This method considers the various entity relations as a single relation. We experiment using the method to confirm the importance of the diversity of predicates.

Fig. 5 portrays the performance for several prediction methods, by looking the chart, we can say that our method outperforms baselines because is RMS error is less than the baselines over three dataset. RMS error is 56.3% of WV for `Person`. SR works better than WV. LP-1P does not work well. The result shows that considering multiple entity relations is important.

Next, to confirm the diversity of predicates contributing to reduction of the RMS value, we perform Like prediction by varying the count of the predicates. Fig. 6 portraits the performance for varying counts of predicates. An increased count of predicates improves the performance. Because we use RDF, we easily increase predicates. Connecting other data source such as GeoNames and Falcon might improve the performance as well.

5.2 Important Predicates for People

We present the coefficient of features in LPP in Table 3, which shows that the logarithm of summation of Likes of occupation is mostly effective. Average Likes of the parents (father and mother) constitute the secondary most effective feature. We can interpret the feature is "inheritance of Likes from parents." Next to parents, logarithm of max Likes of award is effective. This fact supports that the more awards a person achieves, the greater the contribution to his personal brand. From this result, we can construct a Like prediction function.

log-likes = 0.753 log-sum-likes(dbo:occupation) +
0.697 avg-likes(dbo:parent) + 0.640 log-max-likes(dbo:award).

We ranked around 400 predicates. In such predicates, we show the most 10 influential predicates for person in Table 4. Predicates in the table are sorted by their influence.

The table shows that the most influential predicate for person is occupation: the work of the person is effective. The next predicate is parents. We can say that if parents are famous, then the children are also famous. This effect is applicable to various people such as musicians, book authors, and politicians. An award also affects popularity. Therefore, winning a popular award improves the personal brand. Actually, in our community, WWW also chooses a best paper for some papers. This also affects the popularity of the paper. Hometown affects popularity. A person born in a famous place such as New York or Washington D.C. has some advantages. Both influence and influenced-by relationships have

Table 3: The 10 most effective features for people.

Feature	Coefficient
log-sum-likes(dbo:occupation)	0.753
avg-likes(dbo:parent)	0.697
log-max-likes(dbo:award)	0.640
sqrt-sum-likes(dbo:occupation)	0.609
sqrt(dbo:networth)	0.590
log-avg-likes(dbo:parent)	0.563
sum-likes(dbo:spouse)	0.532
sum-likes(dbo:parent)	0.528
max-likes(dbo:parent)	0.511
sqrt-sum-likes(dbo:influenced)	0.499

Table 4: The 10 most influential predicates for people.

Predicate	Influence
dbo:occupation	2.88
dbo:parent	2.52
dbo:award	2.48
dbo:hometown	2.25
dbo:spouse	1.75
dbo:influenced	1.51
dbo:influenced/reverse	1.39
dbo:education	1.30
dbo:relative	1.17
dbo:militaryBranch	1.10

high transmissibility. For that reason, the public figures are growing with their mutual influence. This bidirectional effect is apparent in classical musicians.

Next predicate, dbo:education is the kind of university. That is a person who received a good education becomes popular. This case is apparent for politicians and scientists.

Therefore, we find that popularity is inherited by people from people. If one wants to become popular, making a network link to popular person would be good strategy.

5.3 Mutual Influence Network

Next, we portray the obtained influences as a network. MIN, which represents Like influences among various classes such as not only persons but also places, books, and albums in Fig. 7. A node represents a class which has URI as same as the label of node. The green ones denote person class and subclasses. Thickness of an edge represents the influence $\|\mathbf{u}_p\|_{L1}$ of the corresponding predicate p (see Section 4.3 for definition of $\mathbf{u}_p$). We generate the network based on the ontology of DBPedia: we use the information of domain and range of predicate. If there are triples; then we make a labeled edge and weight it based on the influence.

As discussed in Section 5.2, numerous edges exist with respect to the person class. Many are not shown in the figure.

The original obtained graph is too large. Therefore, we choose two meaningful sub-networks. Fig. 7 shows the sub-network related to work. The center green node is the person class. The Like of a band affects its members, and the Like of Band is transmitted by its single music selections. Like of a single is determined by the artist and its Like is determined by their overall work. For example, Likes of the band members of Beatles depend on the following.

Class `:Work` is the class of all works. The Like of each work transmits to its author, creator, writer and even publisher. Likes of each work is affected by other works. Predicates `:previousWork` and `:subsequentWork` means the order relation of works and it makes list structure. Likes of class :Book influences to its cover artist, illustrator and especially writer. For example, the author of Harry Potter is now attracting many Likes. This fact supports the first assumption. Likes of a scientist are affected strongly by the scientist's doctoral student and notable students. For example, a student in a popular laboratory tends to attract fans in the researcher community.

6. RELATED WORK

Some work related to this paper was conducted earlier. We particularly introduce those studies.

6.1 Prediction

Link-based prediction: Several past studies cope with problems of predicting attributes with regard to given entity from the relations around the entity. Chang *et al.* [3] predict users' locations in Twitter based on social network. The experimentally obtained results dealing with social network improve the accuracy. Lu *et al*. [8] employ entity relations to classification problems. They proposed link-based classification and predict the class of given entity as the function of class of neighbors of entity. Similar to that approach, Hu *et al*. [6] bring a Wikipedia entity relation into a user query intent-classification problem in information retrieval. They first define the ground truth of the user intent class onto a few seed entities and propagate the class into related entities. In contrast to these approaches, the novelty of our approach is disguising multiple types of relationships using predicates. The experimentally obtained results shows agreement with predicates, which contributes to the model's good performance.

Feature construction from ontology: In this paper, we explained automatic feature construction from DBPedia. Paulheim *et al*. [11] provide a feature constructor from Linked Open Data: FeGeLOD. It automatically generates SPARQL, and performs it to RDF data and obtains features. However, the approach does not consider multiple objects with regard to a subject and a predicate exist. This case occurs in feature construction from DBPedia frequently. For example, `db:parent` would take two objects. In this paper, we consider multiple objects and proposed aggregation.

6.2 Entity Linking

Entity linking is an important task of our study that creates a bridge between Facebook and DBPedia. Some approaches have been proposed. Shen *et al.* [12] proposed *LINDEN*, which connects named entities with a knowledge base. They proposed a probabilistic model using description text in addition to title to solve the disambiguation problem. Although we can apply this method to this issue, the difference of the approach presented herein from their approach is that it uses category information provided both by Facebook and by DBPedia. As described in this paper, we show that category information improves the accuracy of the linker. Demartini *et al*. [4] proposed *ZenCrowd*, which leverages a crowd-sourcing platform to solve complex linking tasks. They combined the automatic entity linking method with one done manually by setting minimal confidence to an automatic linker and throwing the task to a crowd-sourcing platform when confidence of the linker falls below the minimal confidence level. That framework can be combined with our approach. Milne *et al*. [9] proposed a knowledge base search engine, *Koru*, which integrates search results and DBPedia and enables users to expand queries. Our Like information would improve their search result ordering algorithm.

6.3 Facebook

This paper is the first describing analysis of Like counts on a large scale. We introduce some earlier reports related to Facebook.

Crawling: The crawling approach for Facebook was proposed [10] with simple crawling approaches such as breadth first search and random walk, which tend to misestimate the original degree distribution. For instance, random walk overestimates values

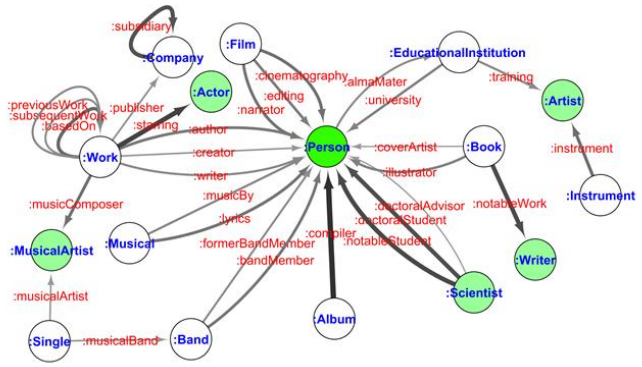


Figure 7: Mutual influence network.

because it samples nodes that have a larger degree. To cope with this erroneous estimation, they proposed two unbiased crawling approaches over a social network: *re-weighting random walk* and the *Metropolis–Hastings random walk*. They conducted each method over the Facebook social graph and compared its degree of distribution to that obtained by uniform sampling. They show for each method recurrence with the original degree distribution. The social networks they obtained using the crawling methods are published on their website. However, both crawling methods assume a structure by which each object is connected by some link such as friendship. Facebook pages have no mutually connective edge. Therefore, we were unable to apply these methods directly. Boshmaf *et al.* [2] conducted a unique crawling method. They set a social network bot on Facebook sending friend requests to ordinary Facebook users automatically. Then they collected information from users who accepted the request. However, this method is forbidden in the Facebook terms of service. In contrast, we conduct a polite crawling method using Facebook Graph API, which is officially provided.

Statistics: One study [1] investigated *Facebook applications*, a feature of Facebook. The key difference of Facebook applications from others is that they fetch information of registered users via Facebook API and provide personalized experiments to users. They carefully analyzed three applications used by eight million users and found communities with a high degree of internal interaction and lower external interaction.

7. CONCLUSIONS

We introduced Like prediction to model Likes of Facebook using Linked Data. We first showed the focused crawler for Facebook, which crawled one-fourth of all Pages. Musicians are popular and obtain many Likes. Like counts show a power-law distribution. Then we gave a formal definition of LPP and we propose the method for the problem. The experiment shows that we can solve LPP precisely with 0.810 RMS. MIN can provide what might increase the Like counts of public figures.

Like prediction can predict the Likes received by an entity. Although we have explained this issue using Facebook-specific terms, this method is widely compatible for other social media such as Twitter because of the flexibility of RDF. Linking Twitter user accounts to Linked Data based on the name and description and learning Like influence model would be necessary to apply this method to Twitter (e.g. predicting follower count). In this case, Like corresponds to followers count. Although our entity linking method is simple, it would be improved using past research.

Combining social media into Linked Data in RDF framework has new opportunity of investigation. In expert search, it is important to ascertain personal skills and expertise. Because several social media provides structured data through an API, we can readily perform such integration by presetting the common property (e.g. description of profile and `rdf:comment`). Identifying effective entity relations based on Like values might open a new area for research. It can reduce information of semantic network. Future investigations might reveal an effective inference algorithm over a semantic network.

REFERENCES

[1] N. Atif, R. Saqib and C. Chen-Nee. Unveiling Facebook: A Measurement Study of Social Network Based Applications. In *Proceedings of the Internet Measurement Conference (IMC)*, 2008.

[2] Y. Boshmaf, I. Muslukhov, K. Beznosov and M. Ripeanu. The Socialbot Network: When Bots Socialize for Fame and Money. In *Proceedings of the 27th Annual Computer Security Applications Conference*, 2011.

[3] Z. Cheng, J. Caverlee and K. Lee. You Are Where You Tweet: A Content-Based Approach to Geo-locating Twitter Users. *In Proceedings of the 19th ACM international conference on Information and Knowledge Management (CIKM2010)*, 2010

[4] G. Demartini, D. E. Difallah, and P. Cudre-Maurou. ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the 21st International World Wide Web Conference (WWW2012),* 2012.

[5] C. F. Hofacker and J. Murphy. World Wide Web banner advertisement copy testing. *European Journal of Marketing*. Vol. 32 Issue: 7/8, pp.703 - 712 doi:10.1108/03090569810224092.

[6] J. Hu, G. Wang, F. Lochovsky, J.T. Sun and Z. Chen. Understanding User's Query Intent with Wikipedia. In *Proceedings of the 19th International World Wide Web Conference (WWW2009),* 2009.

[7] D. J. Lair, K. Sullivan and G. Cheney. Marketization and the Recasting of the Professional Self. *Management Communication Quarterly February*, 2005 vol. 18 no. 3 307-343, doi: 10.1177/0893318904270744

[8] Q. Lu and L. Getoor. Link-based Classification. In *Proceedings of 20th International Conference on Machine Learning (ICML2003)*, 2003

[9] D. Milne, I. H. Witten and D. M. Nichol. A Knowledge-Based Search Engine Powered by DBPedia. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management (CIKM 2007)*, 2007.

[10] G. Minas, K. Maciej, T. B. Carter and M. Athina. Walking in Facebook: A Case Study of Unbiased Sampling of OSNs. In *Proceedings of the 29th Conference on Computer Communications (INFOCOM 2010)*, 2010.

[11] H. Paulheim and J. Fürnkranz. Unsupervised generation of data mining features from linked open data. In *Proceedings of the Second International Conference on Web Intelligence, Mining and Semantics (WIMS2012),* 2012

[12] W. Shen, J. Wang, P. Luo and M. Wang. LINDEN: Lining Named Entities with Knowledge Base via Semantic Knowledge. In *Proceedings of the 21st International World Wide Web Conference (WWW2012),* 2012.

[13] World Wide Web Consortium. Resource Description Framework (RDF) Model and Syntax Specification, 1999.