Personal Name Classification in Web Queries

Dou Shen†, Toby Walker†, Zijian Zheng†, Qiang Yang‡, Ying Li†

†Microsoft Corporation,
One Microsoft Way, Redmond, WA, USA
{doushen, towalker, zijianz, yingli}@microsoft.com

‡Department of Computer Science and Engineering
Hong Kong University of Science and Technology
qyang@cse.ust.hk

ABSTRACT

Personal names are an important kind of Web queries in Web search, and yet they are special in many ways. Strategies for retrieving information on personal names should therefore be different from the strategies for other types of queries. To improve the search quality for personal names, a first step is to detect whether a query is a personal name. Despite the importance of this problem, relatively little previous research has been done on this topic. Since Web queries are usually short, conventional supervised machine-learning algorithms cannot be applied directly. An alternative is to apply some heuristic rules coupled with name-term dictionaries. However, when the dictionaries are small, this method tends to make false negatives; when the dictionaries are large, it tends to generate false positives. A more serious problem is that this method cannot provide a good trade-off between precision and recall. To solve these problems, we propose an approach based on the construction of probabilistic name-term dictionaries and personal name grammars, and use this algorithm to predict the probability of a query to be a personal name. In this paper, we develop four different methods for building probabilistic name-term dictionaries in which a term is assigned with a probability value of the term being a name term. We compared our approach with baseline algorithms such as dictionary-based look-up methods and supervised classification algorithms including logistic regression and SVM on some manually labeled test sets. The results validate the effectiveness of our approach, whose *F1* value is more than 79.8%, which outperforms the best baseline by more than 11.3%.

Categories and Subject Descriptors

I.5.4 [**Pattern Recognition**]: Applications—*Text processing*; H.4.m [**Information Systems**]: Miscellaneous

WSDM'08, February 11–12, 2008, Palo Alto, California, USA.

General Terms

Algorithms, Experimentation, Verification.

Keywords

Web query, web search, personal name classification, probabilistic dictionaries

1. INTRODUCTION

Personal names are an important kind of Web queries in Web search. By our estimation on the data collected from a major commercial search engine, there are 2∼4% daily Web queries in the form of exact personal names. The count is even higher if we consider all queries that contain personal names, which is about 30% according to [2]. The count is almost doubled in the switching queries, which refer to the queries for which the Web users cannot get satisfying search results and switch to another search engine to try the queries again. The difference of the numbers reflects the fact that Web users often are unsatisfactory towards personal name search results. Besides the quantitative fact to show the importance of personal name queries, another well-known fact that makes personal name queries important, especially to commercial search engines, is that most Web users tend to put their names to a search engine and judge the search engine's performance. Therefore, from both technical and commercial view, we have to improve the search results for personal name queries.

In general, once we know a query is a personal name, we can apply some specific algorithms by considering some special features such as whether a page is a personal hompepage and the proximity of the query terms in the page. Therefore, to improve the search performance on personal names, the first step is to design an algorithm for personal name classification in Web queries, that is to decide whether a query is a personal name. There are also many other additional benefits of accurate personal name classification. One typical example is online advertisement suggestion. For example, given a query "toby walker", it may be less valuable to return the advertisement like "The Walking Aid Store", since the meaning of "walker" in the name context is not related to "walk" any more.

There is some work from different computer science fields related to personal names. Some of them work on personal name extraction from documents or name list construction from internet, while others work on personal name disam-

biguation in academic citations or name resolution in Web search [4, 22]. However such work is fundamentally different from the problem of personal name classification in Web queries. In previous work, there is abundant and valuable contextual information embedded in the document or search results. For example, in "Dr. Yang's student, Dou Shen", the text "Dr. Yang's student" provides strong evidence that "Dou Shen" is a personal name. Conversely, the information conveyed by Web queries is quite sparse, with only 2.35 terms in a Web query on average [18]. To solve the problem of information sparseness in Web queries, some query enrichment methods through search engines have been applied [17]. However, it is quite time-consuming to collect context through search engines for online, real-time applications. So far as we know, this paper is the first attempt to work on the Web queries directly to decide whether a query is a personal name.

Because of the sparse information embedded in Web queries, it is difficult to obtain sufficient features for applying some conventional classification algorithms, which have been applied successfully in document and Web page classification [24]. An alternative and straight-forward solution for the problem of personal name classification in Web queries is using dictionary look-up. Given a query, we can check whether the contained terms are in some predefined name term dictionaries. However, when the dictionaries are small, this method tends to make false negatives, that is, missing some correct personal names such as "megan gianchetta". On the other side, when the dictionaries are large, it tends to generate false positives by classifying non-names as names such as "rotten stone". What is more, it is hard for this method to trade-off precision and recall such that it cannot cater for different application scenarios with different requirements.

To solve this problem, we put forward an approach based on the construction of probabilistic name-term dictionaries to generate probabilistic outputs, so that we can trade-off precision and recall. In our approach, given a list of candidate name terms, we try several different methods for building probabilistic name-term dictionaries in which each term comes with a number to show the probability of the term being a first-name term (or last-name term). We also study how to construct the probabilistic dictionaries if we do not have a proper candidate name term list. After obtaining the probabilistic dictionaries, we can calculate the probability of a query being a personal name based on some personal name grammars.

In this paper, to validate our algorithms, we compare our proposed method with two kinds of baselines: supervised classification methods and dictionary based look up. The experimental results on some human-labeled data sets validate the effectiveness of our proposed method.

The rest of the paper is organized as follows. In Section 2, we present some previous work related to personal name and Web query analysis. We refine our problem description in Section 3. Section 4 describes our proposed method. We describe some baselines in Section 5 and study the performance of our method and the comparison with baselines in Section 6. Finally, we conclude our work in Section 7.

2. RELATED WORK

Personal names are an specific and important kind of entities in Web queries. More and more attention has been attracted to personal name search or people search. In [8], Dozier studies some specific strategies for personal name search, which is proved to be more effective than using the strategies for common queries. Besides that, there is plenty of work on personal name disambiguation or resolution in Web searches. In [2], the authors provide a testbed for people searching strategies. [22] describes a person resolution system in person search, which can cluster the returned Web pages for a personal name query. All such work assumes that it is known that the queries are personal names. However, it is not trivial to judge whether a query is a personal name in real applications.

Some related research has been conducted on personal names recognition/extraction from documents. For example, in [5], Chen et al. use a statistical approach to extracting personal names from a corpus. Their approach can both automatically learn the characteristics of personal names from a large training corpus and make good use of human empirical knowledge (in terms of Context Free Grammar). Personal names, as a special case of named entities, have been widely studied in the literature of named entity recognition (NER) [9, 6]. [9] presents a classifier-combination experimental framework for NER in which four diverse classifiers (robust linear classifier, maximum entropy, transformation-based learning, and hidden Markov model) are combined under different conditions. In [6], Leong et al. present a maximum entropy approach for NER tasks, where NER not only make use of local context within a sentence, but also make use of other occurrences of each word within the same document to extract useful features (global features). Similar idea to leverage contexts globally is validated in [12]. In [4], the authors propose a method to extract proper names and their associated information from web pages, which are further used to construct white pages for Internet/Intranet users or to build databases for finding people and organizations on the Internet. All of these methods rely on the rich contextual information for personal name recognition. However, to recognize personal names in Web queries, we do not have such information.

Another group of related work is the research conducted on Web queries, where different kinds of information has been exploited to represent Web queries. In [13], Lee et al. work on an important problem to detect user's goal through their submitted queries. The queries are categorized as navigational queries or informational queries and their method is based two types of features including user-click behavior and anchor-link distribution. In [17], we study the query classification problem and solve the data sparse problem of Web queries by enriching them through search engines. The similar idea is exploited in [7], where the authors predict the commercial intension of Web queries. There is also some work on clustering Web queries using clickthough data in unsupervised manners [23, 3]. We can see that all of these work either uses clickthrough data or uses search engines to enrich Web queries. However, we cannot rely on the former method for newly emerging queries and the latter method is time consuming which is not applicable for online applications. Therefore, in this paper, we put forward a method working in two stages. In the offline stage, we exploit several kinds of resources to construct probabilistic dictionaries. In the online stage, we rely on some personal name grammars to classify each query directly so that we can process each query very efficiently.

3. PROBLEM DEFINITION

As we discussed in Section 1, personal names are an important kind of Web queries. In this paper, we focus on the most basic problem of detecting whether a Web query in whole is a personal name. To make our problem clear, let us define "personal name" stringently. According to the definition from Wikipedia, "A name is a label for a person, thing, place, product (as in a brand name) and even an idea or concept, normally used to distinguish one from another" (http://en.wikipedia.org/wiki/Name). Then it is easy to define a personal name as a label for a person. Since different cultures have different conventions for personal names (http://en.wikipedia.org/wiki/Personal_name), in this paper, we focus on English personal names. In this paper, for a query in whole being a personal name, it is allowed to contain title terms (such as "dr") and suffix terms (such as "sr"), but it is not allowed to contain other kinds of terms. For example, both the queries, "john smith" and "dr. john smith" are personal names while the query "john smith pictures" is not a personal name.

The solution of the above basic problem can be extended to other related problems easily. For example, it can be used to detect whether a query contains a personal name (such as "john smith pictures") and whether a string is a concatenation of terms in a personal name (such as "johnsmith"). In this paper, we only study the basic problem, that is whether an entire query is a personal name.

4. OUR SOLUTION

In this paper, we put forward an approach for personal name classification in Web queries based on the construction of probabilistic name-term dictionaries and personal name grammars. Figure 1 provides an overview of our approach, which works under two stages: offline stage and online stage. The offline stage is for probabilistic dictionary construction. Given a list of candidate name terms, we use several different methods to estimate the probability of each candidate term being a first-name term (or last-name term). We also study how to construct the probabilistic dictionaries if we do not have a large candidate name term list. The online stage is for classification, in which we can calculate the probability of a query being a personal name based on the constructed probabilistic dictionaries and some grammars. Our approach is expected to exploit all possible resources effectively in the offline stage without concerning time complexity and classify Web queries efficiently in the online stage.

4.1 Constructing Probabilistic Dictionaries

4.1.1 Concept Definition

Given a candidate term list, our goal is to construct a probabilistic dictionary, in which each term has a number to show its probability to be a first-name term or last-name term. Before coming to the details of the methods for constructing probabilistic dictionaries, we define several concepts for the convenience of description:

Candidate Dictionary: a candidate dictionary contains a list of candidate terms for which we need to estimate their probabilities being first-name terms or last-name terms;

Term Context: a term context is a piece of text in which a term shows up, which reflects the context of the term's occurrence. For example, both "... toby walker ..." and "... city of walker..." are the term contexts of "walker". One way to obtain the term context is through search engines. By submitting a term as a query to a search engine, we can get a list of returned Web pages which contain the term, as well as titles and snippets to show how the term is used in those pages. Therefore, we can take the titles and the snippets as the term context of the target term.

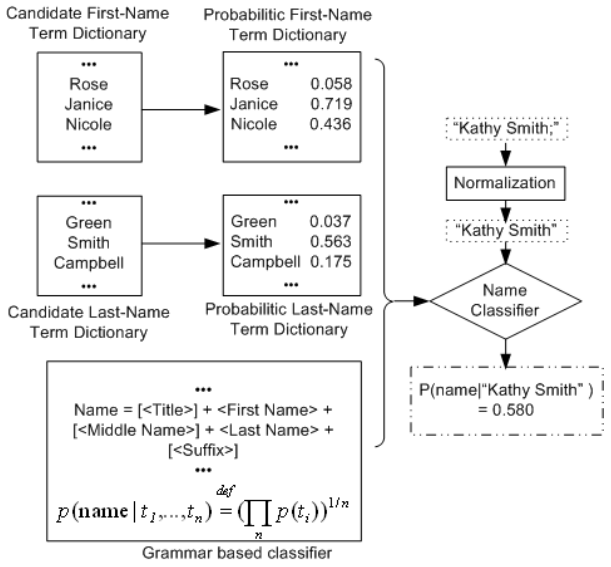

Figure 1: Overview of the name classification method.

Name Term Context: a name term context is the context in which a term acts as a first-name term or last-name term; for example, in the above example, "... toby walker ..." is a name term context for "walker" while "... city of walker..." is not.

Name Context: a name context is a piece of text in which a personal name shows up. For example, "... dr. qiang yang's student ..." is a name context for the personal name "dr. qiang yang". Similar to term context, we can use search engines to obtain the name context for a certain personal name.

4.1.2 Relative Frequency (RF)

In most cases, we can obtain a directory of personal names as well as the occurrence number of each personal name (in case different people have the same name). For example, we can obtain the data from a company or from a census bureau. Then we can split the personal names into first-name terms and last-name terms. At the same time, we can get the number of times for each term being a first-name term or last-name term. Then we can estimate the probability of each term being a first-name term in the name directory using the following equation [8].

$$p(t \text{ is a } first\ name\ term) = \frac{F}{N} \quad (1)$$

where F is the number of occurrence of t as a first-name term; N is the number of names. We can estimate the probability of a term being last-name term in the same way.

4.1.3 Context Probability (CP)

It is reasonable to assume that if a term acts as name term with high probability, its term context should be a personal name context with high probability. Therefore, we can estimate the probability of a term being a name term by estimating the probability of its term context being generated by personal name contexts. In this paper, we collect the name contexts based on a set of personal names through search engines and then build a unigram model [14] to represent the name contexts. Given a term and its term contexts, we calculate the probability of each term context being a name context using the unigram model. After that, we estimate the probability of a term being a name term by averaging the calculated probabilities of its corresponding term contexts.

4.1.4 Co-occurrence in Snippet (S-coOcc)

As discussed in Section 4.1.1, we can obtain the term context of a given term through search engines. Then, if we can detect name term contexts among these term contexts, we can estimate the probability of the term being a name term by calculating the relative frequency of the name term contexts compared to the term contexts. In this paper, we define several scenarios based on which we can regard some term contexts as name term contexts. For a candidate first name term, such as "john", we assume in the following contexts, it is acting as a first name term: (1) it is followed by a last name term, such as "john smith"; (2) it is followed a first name term and then a last name term, such as "john maynard smith"; (3) it is followed by a special kind of verbs such as "did, said, announced, claimed ..." as in "john said"; (4) it is followed by "'s", such as "john's ...". For a candidate last name term, such as "smith", we assume it is acting as a last name term in the following contexts: (1) it is preceded by a first name term; (2) it is preceded by a title such as "Dr. Smith"; (3) it is preceded by a letter such as "J Smith" or "J. Smith". To define the first two scenarios for the candidate first name terms and the first scenario for the candidate last name terms, we need some "golden standard" dictionaries to decide whether a term is a first-name term or last-name term. The "golden standard" dictionaries can be much smaller than the candidate dictionaries. However, they should be purer and do not include non-name terms. Otherwise we will introduce significant noise into our probability estimation. In the experimental section, we will study the effect of the "golden standard" dictionary thoroughly. Given a term t, with both the term contexts and name term contexts available, we can estimate its probability being a name term according to equation (2) under the maximum likelihood criteria.

$$p(t \text{ is a name term}) = \frac{\#name\_Occ}{\#all\_Occ} \quad (2)$$

where $\#all\_Occ$ denotes the number of all occurrences of t in its corresponding term contexts; $\#name\_Occ$ is the number of its occurrence in name contexts.

Although the way to define name term contexts is intuitive and straightforward, it is proved to be effective in our empirical study. To provide more precise ways for defining name term contexts is left for our future work.

4.1.5 Co-occurrence in Bigrams (B-coOcc)

Given a term, using Search engines to get the term contexts, we usually consider only the top returned results for computational simplicity, which are some typical examples of all the term contexts. In order to leverage all possible term contexts, another way to get the context is from occurrence statistics of n-grams collected from a large corpus. In this paper, we use bigram statistics collected from all the indexed Web pages by an commercial search engine. Several samples of the bigrams are shown in Table 1. For each row, the first number is the occurrence count of the bigram in the corpus. The second number is the number of Web pages containing the bigram. Based on the similar method as used by S-coOcc, we can get the probability of each candidate name term being a first-name term or last-name term.

Table 1: Examples of entries in a bigram list

...			
2227	2173	toby	up
1250	758	toby	wachter
2307	1304	toby	walker
1013	358	toby	walking
1178	689	toby	walsh
18782	12951	toby	was
2244	1876	toby	we
...			

Algorithm 1 : Enlarge Candidate Dictionaries

INPUT:

i. Original candidate dictionaries for first-name terms and last name terms: FD_0 and LD_0;

ii. Thresholds of probability for a term being a first name term (θ_F) and last-name term (θ_L);

iii. Iteration number: N;

OUTPUT:

Enlarged candidate dictionaries with the probability of each term being a name term;

FOR i = 0 : N

1. Estimate the probability of each term in FD_i being a first-name term;
2. Estimate the probability of each term in LD_i being a last-name term;
3. HPFT = $\{t|t \in FD_i$ and $p(t$ is a first-name term) $> \theta_F\}$
4. HPLT = $\{t|t \in LD_i$ and $p(t$ is a last-name term) $> \theta_L\}$
5. Collect term contexts for the terms in HPFT and HPLT;
6. $FD_{i+1} = FD_i \cup \{t|t$ is followed by a term from HPLT in its corresponding term contexts$\}$
7. $LD_{i+1} = LD_i \cup \{t|t$ is preceded by a term from HPFT in its corresponding term contexts$\}$

END FOR

4.1.6 Enlarge Candidate Dictionaries Automatically

As we will validate in our experiments, a large candidate dictionary with high coverage and less noise is beneficial to

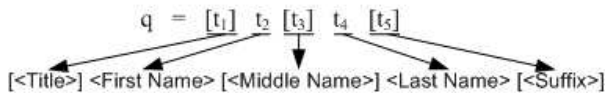

Figure 2: Illustration of parsing queries using personal name grammars.

the personal name classification problem. However, it is not easy to obtain such candidate dictionaries. In this section, we propose an algorithm to start from small candidate dictionaries and enlarge the dictionaries automatically. The algorithm is presented in Algorithm 1. In steps 1 and 2, we estimate the probability of each term in the current candidate dictionaries; in steps 3 and 4, we select the terms in the current candidate dictionaries whose probabilities are larger than the predefined thresholds and denote them by HPFD and HPLD; in step 5, we collect the term contexts of the terms in HPFD and HPLD; lastly, among the collected term contexts, we select the potential candidate name terms and add them to the current candidate dictionaries; after that, we start a new iteration until the iteration number reaches the predefined number.

4.2 Grammar Based Classifier

In this paper, we focus on English personal names. Following are the major grammars to parse a personal name:

*<**Personal Name**> ::= [<Title>] <First Name>[<Middle Name>]<Last Name>[<Suffix>]*

*<**Title**> ::= dr || doctor || ms, ...*

*<**First Name**> ::= term1 || term1−term2 where term1 and term2 are from a first-name term dictionary;*

*<**Last Name**> ::= term1 || term1−term2 where term1 and term2 are from a last-name term dictionary;*

*<**Middle Name**> ::= <First Name> || <Last Name>;*

*<**Suffix**> ::= sr.|| jr.|| III, ...*

Note that for <First name> or <Last name>, it can be two terms connected by a hyphen, such as "bowen-mccombs".

Given a query $q = t_1 \dots t_n$, we use the above grammars to parse the query as illustrated in Figure 2. Then we can use the following equation to calculate the probability of q being a personal name:

$$p(q \text{ is a personal name}) = (\Pi_{i=1}^{n} p(t_i))^{1/n} \quad (3)$$

If a term t is a title term or suffix term, we assign $p(t) = 1.0$. By this way, we can improve the probability of a query containing title and/or suffix terms. When a term is a combination of two terms connected by a hyphen, such as "bowen-mccombs", if it is not in the probabilistic dictionaries, we define its probability as the average of the probabilities of the two single terms.

Equation (3) is actually a geometric mean of the probability of each term. An alternative way is to use arithmetic means. However, geometric means are better at penalizing the terms with low probabilities than arithmetic means. What is more, our empirical study shows that geometric means work better in our problem. Therefore, we adopt geometric means in this paper.

5. BASELINE METHODS

For the personal name classification problem, there are two kinds of intuitive and straightforward methods. The first one is a boolean method based on dictionary look-up. The other is to treat the personal name classification problem as a conventional classification problem. As we discussed in the introduction section. Both of these methods have their shortcomings. We present a brief introduction of these methods and then compare these methods with our proposed method empirically in next section.

5.1 Boolean method by Dictionary Look-up

This method can be regarded as a simplified version of our probabilistic grammar based method. Given a query, we can parse it according the personal name grammars given in Section 4.2. After that we look up the candidate dictionaries to see whether the terms in the query correspond to a first-name term and a last-name term correctly. If correct, the query is classified as a personal name. For example, given a query "john smith", if "john" is in the first-name term dictionary and "smith" is in the last-name term dictionary, "john smith" is classified as a personal name. Otherwise, it is not. When the candidate dictionaries are small, this method tends to make false negatives, that is, missing some correct personal names. On the other side, when the dictionaries are large, it tends to generate false positives by classifying non-names as names. What is more, this method cannot trade-off precision and recall.

5.2 Supervised Methods

Besides classifying a query in an unsupervised way as shown in the above methods, we can treat the problem of personal name classification in Web queries as a conventional binary classification problem. The two most important aspects of the supervised methods are query representation and classification algorithms. In our paper, we represent each query by a vector of features and use either Logistic Regression Models or Support Vector Machines as the classification algorithm.

5.2.1 Features

In this paper, we extract dozens of features for each query. Following list some representative features for a query.

f1: the number of terms in the query;
f2: whether the query contains a title term;
f3: whether the query contains a suffix term;
f4: whether a term is in the first-name term dictionary of CENSUS;
f5: whether a term is in the last-name term dictionary of CENSUS;
f6: the probability of a term being a first name term according to the probabilistic dictionaries constructed by us;
f7: the probability of a term being a last name term according to the probabilistic dictionaries constructed by us;
f8: the probability of a term being generated by a character level bigram model [15] trained on first-name terms (the terms which are in the WP first-name term dictionary);
f9: the probability of a term being generated by a character level bigram model trained on last-name terms (the terms which are in the WP last-name term dictionary);
f10: the probability of a term being generated by a character level bigram model trained on general terms (the terms which are included in a collection of bigram data but not in the WP dictionaries);

f11: whether a term starts with a capital character.

Note that we count on the CENSUS dictionaries, WP dictionaries and bigram data to define features *f4*, *f5*, *f8*, *f9*, *f10*. All of these data sets will be introduced in Section 6.1.1.

5.2.2 Classification Algorithms

Logistic Regression Models (LR)

Logistic regression models are also called maximum entropy models, and are equivalent to single layer neural networks that are trained to minimize entropy [1]. It has been widely used for handling binary class variables [19, 10]. The logistic regression model can be considered as a generalized linear model by making a logistic transformation. Instead of predicting the precise numerical value of a dependent variable as in linear models, the logistic regression model predicts the probability of one class being true. Specifically, given a query q and its features $X = x_1, \ldots, x_n$, the logistic regression model takes the following form:

$$p(q \text{ is a personal name}|X) = \frac{e^{\beta_0+\beta_1 x_1+\ldots+\beta_n x_n}}{1+e^{\beta_0+\beta_1 x_1+\ldots+\beta_n x_n}} = \frac{e^{\beta X}}{1+e^{\beta X}} \quad (4)$$

where $\beta = (\beta_0, \beta_1, \ldots, \beta_n)$ are parameters reflecting the relative importance of the features. The parameters are usually estimated by maximum likelihood.

Support vector machine (SVM)

Support vector machine (SVM) is a powerful learning method introduced by V.Vapnik et al. [21]. It is well founded in terms of computational learning theory and has been successfully applied to many fields [11]. SVM operates by finding a hyper-surface in the space of possible inputs. The hyper-surface attempts to split the positive examples from the negative examples by maximizing the distance between the nearest of the positive and negative examples to the hyper-surface. Intuitively, this makes the classification correct for testing data that is near but not identical to the training data. There are various online toolbox to train SVMs. In this paper, we use the SVM^{light} software package (http://svmlight.joachims.org/). For the comparison convince with other methods, we utilize the method in [16] to convert the output to a probability.

6. EXPERIMENTS

6.1 Data preparation

6.1.1 Dictionaries

Golden dictionaries and candidate dictionaries are key resources to our proposed method. In this paper, we collect the dictionaries from three different resources. The first resource is "Census Bureau"[1], from which we get a first-name dictionary with 5,494 terms and a last-name dictionary with 88,799 terms. The second resource is from DBLP, a computer science bibliography website[2]. By extracting the author names from the bibliography, we get a first-name dictionary with 64,187 terms and a last-name dictionary with 167,965 terms. The third resource is "phone-book white pages", an internal resource used in Microsoft. From this resource, we extract a first-name dictionary with 762,905 terms and a last-name dictionary with 2,045,637 terms. For convenience, we denote the dictionaries from "Census Bureau" as "CENSUS", the dictionaries from DBLP as "DBLP" and the dictionaries from the "phone-book white pages" as "WP". Table 2 summarizes the sizes of different dictionaries. Table 3 and Table 4 show the difference of the different kinds of dictionaries. The number in each cell reflects the number of terms which are in the dictionary indexed by the row but not in the dictionary indexed by the column. We can see that the WP dictionaries are much larger than the CENSUS dictionaries. However, the CENSUS dictionaries are relative pure while the WP dictionaries are quite noisy. For example, some terms such as "University", "Office", "Apartment" which are obviously not name terms show up in the WP dictionaries. In order to remove the impact of the noises in the WP dictionaries while keep their advantage being comprehensive, we propose to construct probabilistic dictionaries.

[1]http://www.census.gov/genealogy/names/

[2]http://dblp.uni-trier.de/

Table 2: Sizes of the dictionaries from different resources

	DBLP	CENSUS	WP
First Name	64,187	5,494	762,905
Last Name	167,965	88,799	2,045,637

Table 3: Differences in First-Name Terms of Different Kinds of Dictionaries

	DBLP	CENSUS	WP
DBLP	0	60,559	24,841
CENSUS	1,891	0	2
WP	726,292	759,642	0

Table 4: Differences in Last-Name Terms of Different Kinds of Dictionaries

	DBLP	CENSUS	WP
DBLP	0	127,993	41,712
CENSUS	50,625	0	886
WP	1,809,147	1,844,603	0

6.1.2 Name Context

As we introduced in Section 4.1.1, a name context is a piece of text in which a personal name shows up, which reflects the context of the occurrence of a personal name. In this paper, a collection of name contexts are used to train a unigram model which will be further used by the method CP described in Section 4.1.3. To obtain the name contexts, we select the top 200,000 personal names extracted from the "phone-book white pages" which are ordered by their occurrence numbers. After that, we submit these personal name to Live Search (http://www.live.com) and use the top 50 returned pages (including titles and snippets) as the name contexts for each personal name. By this way, we collect about 10,000,000 name contexts in total.

6.1.3 Term Context

A term context reflects the context of the occurrence of a term, as discussed in Section 4.1.1. In this paper, we use two approaches to obtain the term contexts for a term. The first approaches is through Search Engines, similar to the way used to collect name contexts as shown in Section 6.1.2. We submit about 2,000,000 candidate name terms to Live Search and take the top 50 returned pages (including titles and snippets) as the term contexts for each term. By this way, we collect 80,000,000 term contexts in total.

As discussed in 4.1.5, another way to collect the term contexts is to use n-grams collected from a large corpus. In this paper, we use the bigram statistics collected from all the indexed Web pages by Live Search. Table 1 shows several examples of the bigrams.

6.2 Testing Data Sets

We collect two data sets to study our proposed methods. Both of the two data sets are randomly sampled from the query logs in Live Search, with different sizes. One of data sets contains 10,000 queries with 232 queries being personal names. The other data set contains 2,000 queries with 81 queries being personal names. These data sets are labeled by native speakers. We denote these two data sets as 10000-query set and 2000-query set respectively. In this paper, we use the 2000-query set as the validation data set to tune the parameters and use the 10000-query set as the testing data set.

Before applying the classification methods on the data sets, we use a simple normalization method to preprocess the queries by removing the characters such as quotation marks, semicolons.

6.3 Evaluation Metrics

We employ the standard measures to evaluate the performance of personal name classification in Web queries, i.e. precision, recall and F1-measure [20]. Precision (P) is the proportion of actual positive class members returned by the system among all predicted positive class members returned by the system. Recall (R) is the proportion of predicted positive members among all actual positive class members in the data. F1 is the harmonic average of precision and recall which is defined as $F1 = 2PR/(P + R)$.

6.4 Results and Analysis

In this section, we study our proposed methods from several aspects and present the experimental results as well as the analysis. For our probabilistic grammar based method, we need to tune the threshold of the probability for a query being a personal name to get the best performance on a certain data set. In this paper, we use the 2000-query set as a validation data set based on which we obtain the threshold where our method reach the best performance in terms of F1. Then we report the performance of our method on the 10000-query set with the same threshold. For the supervised methods, we use 3-fold cross validation to remove the uncertainty of data split. We randomly split the 10000-query data set into 3 folds and use two folds as training data and the other fold as testing data. In order to trade-off precision and recall and reach the best F1, we need a threshold of probability for the supervised methods too. The threshold is also obtained on the validation data set.

Table 5: Comparison among our method and the baseline methods

	Pre	Rec	F1
Prob.DBLP	0.891	0.457	0.604
Prob.CENSUS	**0.942**	0.487	0.642
Prob.WP	0.779	0.819	**0.798**
Bool.DBLP	0.651	0.466	0.543
Bool.CENSUS	0.803	0.491	0.610
Bool.WP	0.127	**0.884**	0.222
SVM	0.798	0.595	0.681
LR	0.785	0.659	0.717

6.4.1 Comparison Among Our Method and the Baselines

As discussed in section 4.1, we have several different methods to construct probabilistic dictionaries. In this experiment, we adopt S-coOcc. For S-coOcc, we need golden dictionaries and candidate dictionaries, as shown in 4.1.4. Here we use CENSUS dictionaries as the golden dictionary and test the candidate dictionaries including DBLP, CENSUS and WP. "Prob.DBLP" denotes our probabilistic method based on the DBLP dictionaries and "Bool.DBLP" denotes the boolean method based on the DBLP dictionaries. "Prob. CENSUS", "Prob.WP", "Bool.CENSUS", "Bool.WP" can be explained in the same way.

From Table 5 we can see that for all dictionaries, our probabilistic grammar based method can improve the performance significantly in terms of F1. The improvement is 5.2% on CENSUS dictionaries and 259.5% on WP dictionaries. Another observation is that larger dictionaries (WP dictionaries) has obvious advantages in our proposed method, but disadvantages in boolean method based on dictionary look-up. It is because that the noise in the large dictionaries will greatly hurt the performance of dictionary look-up methods. However, our proposed methods can effectively reduce the effect of the noise by estimating their probability being name terms. What is more, the larger the dictionaries, the higher coverage they have. That is why larger dictionaries have advantages in our methods.

Another observation from Table 5 is that our proposed methods is clearly better than the two supervised methods, SVM and LR, by 17.2% and 11.3% in terms of F1 respectively. This observation validate our claim that conventionally successful classification methods may not work well in the problem of personal name classification in Web queries because of the sparse information contained in queries.

6.4.2 Comparison of Different Approaches for Constructing Probabilistic Dictionaries

Table 6 presents the comparison results among the four approaches for constructing probabilistic dictionaries. In this experiment, for S-coOcc and B-coOcc, we use CENSUS dictionaries as the golden dictionaries and use WP dictionaries as the candidate dictionaries. For RF (relative frequency), the data are from the "phone-book white pages". From this table, we can see that S-coOcc achieves the best performance, which is 1.8%, 21.9% and 14.5% higher than the second best method, B-coOcc, in terms of precision, recall and F1 respectively. The reason for the performance difference is that although B-coOcc exploits much more term contexts than S-coOcc, the term contexts used by B-coOcc

Table 6: Comparison of Different Approaches for Constructing Probabilistic Dictionaries

	Pre	Rec	F1
S-coOcc	**0.779**	**0.819**	**0.798**
B-coOcc	0.765	0.672	0.716
RF	0.628	0.349	0.449
CP	0.237	0.569	0.335

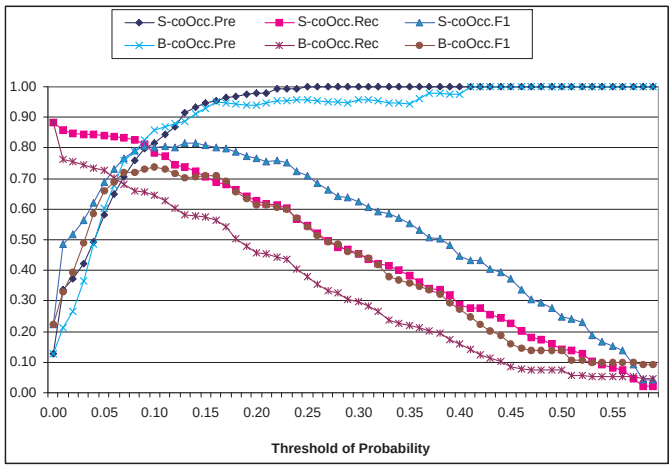

Figure 3: Detailed Results of S-coOcc and B-coOcc along Different Probability Thresholds.

(bigrams) do not provide enough information as those used by S-coOcc. For example, the third scenario defined for first-name terms in Section 4.1.4 is not valid in the bigram data. Therefore, it is beneficial to exploit more complicated term contexts such as trigram data in the future. CP is the worst method and the bad performance may be caused by: (1) Context probability is not a good way to reflect the probability of a term being a name term; (2) CP cannot distinguish first-name terms and last-name terms. Although the method RF is better than CP, it is much worse than S-coOcc and B-coOcc. The reason for the bad performance of RF is that RF just measures the relative frequency of a term being a name term internally (among all name terms) without considering the relative frequency of a term being a name term externally (among all its usages, that is the term contexts defined in Section 4.1.1). For example, although the first name of one author of this paper, "zijian" is not a frequent first-name term in the "phone-book white pages", it has high relative frequency being a first-name term among its term contexts.

Figure 3 provides the detailed performance of S-coOcc and B-coOcc when we change the probability thresholds. We can see that by increasing the probability thresholds, the precision of both methods will be improved clearly while the recall is decreased. The F1 values of both methods get improved at first and then decrease. The best F1 based on S-coOcc can reach 81.6% when the threshold is 0.15.

6.4.3 *Effect of Golden Dictionaries and Candidate Dictionaries*

From previous experiments, we can see that S-coOcc is the best method to construct probabilistic dictionaries. In this

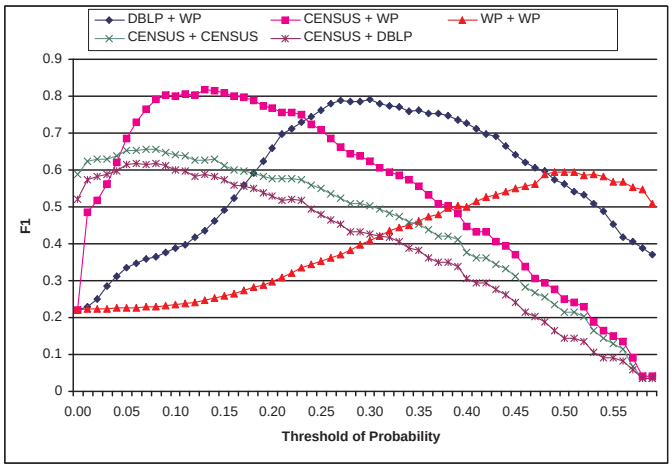

Figure 4: Detailed Comparison of the Different Combinations of Golden Dictionaries and Candidate Dictionaries.

experiment, we use S-coOcc to study the effect of golden dictionaries and candidates dictionaries. Table 7 shows the results for all possible combinations of the dictionaries DBLP, CENSUS and WP being golden dictionaries and candidate dictionaries. The rows of Table 7 are golden dictionaries and the columns of Table 7 are candidates dictionaries. Among all the combinations, we see that we can reach the best F1 when we use CENSUS as the golden dictionaries and use WP as the candidate dictionaries. The reason is that CENSUS are the purest dictionaries (even compared to the DBLP dictionaries) which can provide the best dictionaries to define name term context, while WP are the largest dictionaries which have highest coverage. Therefore, by taking CENSUS as the golden dictionaries and WP as candidate dictionaries, we can keep the advantages of WP with high coverage and reduce the bad impact of the noises in WP.

To make the comparison clearer, Figure 4 shows the F1 values of different combinations of golden dictionaries and candidate dictionaries when we change the probability threshold. In the figure, "DBLP + WP" means that we use DBLP as the golden dictionaries and use WP as the candidate dictionaries. In this experiment, we show the results of the combinations by using the WP (the largest dictionaries) as candidate dictionaries and then vary the golden dictionaries (that is "DBLP +WP", "CENSUS+WP" and "WP +WP") and using CENSUS (the purest dictionaries) as the golden dictionaries and vary the candidate dictionaries (that is "CENSUS + DBLP", "CENSUS + CENSUS", "CENSUS+WP"). Note that the probability threshold for the peak performance of "WP +WP" is much larger than others. The reason is that by using "WP" as the golden dictionaries, more name term contexts will be defined and the overall probability a term being a name term will be increased and then the threshold of the peak performance is increased accordingly.

6.4.4 *Effect of the Sizes of Golden Dictionaries and Argumentation of Golden Dictionaries*

In this section, we further study the effect of the sizes of the golden dictionaries and how to argument the golden dictionaries automatically. In these experiments, we take WP

Table 7: Effect of Golden Dictionaries and Candidate Dictionaries

	Pre			Rec			F1		
	DBLP	Census	WP	DBLP	Census	WP	DBLP	Census	WP
DBLP	0.885	0.894	0.783	0.466	0.509	0.776	0.610	0.648	0.779
Census	0.891	0.942	0.779	0.457	0.487	0.819	0.604	0.642	**0.798**
WP	0.737	0.826	0.604	0.483	0.513	0.586	0.583	0.633	0.595

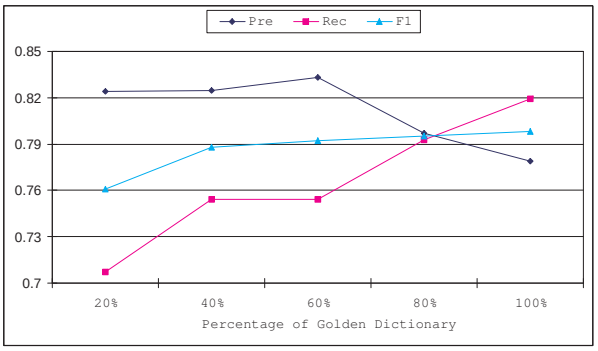

Figure 5: Classification Performance When We Change the Size of Golden Dictionaries.

Table 8: Classification Performance When We Augment Golden Dictionaries Automatically

	0	1	2	3	4	5
Pre	0.779	0.797	0.844	0.948	0.914	0.910
Rec	0.819	0.810	0.772	0.707	0.737	0.724
F1	0.798	0.803	0.806	0.810	**0.816**	0.806

Table 9: Classification Performance When We Enlarge Candidate Dictionaries Automatically

	0	1	2	3	4	5
Pre	0.944	0.802	0.789	0.737	0.737	0.737
Rec	0.366	0.595	0.595	0.603	0.603	0.603
F1	0.528	**0.683**	0.678	0.664	0.664	0.664

as the candidate dictionaries and use S-coOcc to construct the probabilistic dictionaries.

Figure 5 shows the classification results when we change the size of the golden dictionaries by randomly selecting 20%, 40%, ..., 100% terms from CENSUS as the golden dictionaries. From the figure we can see the when the sizes of the golden dictionaries are less than 60% of CENSUS, by increasing the size of the golden dictionaries, all the measurements including precision, recall and F1 are increased steadily. When the sizes are larger than 60%, further increasing the size of golden dictionaries can improve F1 and recall, but will hurt the precision clearly.

Table 8 shows the classification results when we increase the golden dictionaries. Following describes the approach to increase the golden dictionaries. In the initial stage, we take CENSUS as the golden dictionaries and then construct the probabilistic dictionaries. After that, in each iteration we select the top 1,098 terms and 17,760 terms (about 20% of the terms in the corresponding CENSUS dictionaries) from the probabilistic first-name term dictionary and last-name term dictionary and add them to the golden dictionaries respectively. From the table we can see that for the first four iterations, F1 is increased and the F1 is decreased in the fifth iteration. The reason is that pure and large golden dictionaries can define the name term contexts precisely for the method S-coOcc, that is why we can increase the F1 value at first. However, by adding too many words to the original golden dictionaries, we may introduce noise which will hurt the classification performance finally.

6.4.5 *Effect of Enlarging Candidate Dictionaries*

Recall the algorithm described in Section 4.1.6, we need to provide three types of parameters as input. In this experiment, we use B-coOcc for simplicity so that we do not have to rely on Search Engines to get the term context for newly discovered candidate term in each iteration. We use CENSUS as the golden dictionaries and the initial candidate dictionaries. For the two probability thresholds, we set both of them as 0.1 empirically. We will study the effect of the probability threshold in our future work. Table 9 shows the classification performance when we enlarge the candidate dictionaries iteratively. We can see a significant achievement in the first iteration where the F1 is improved by 29.4% relatively, which is near the performance (0.716 as shown in Table 6) when we use WP as the candidate dictionaries. However, the performance becomes a little worse and then stable when we run more iterations. This observation can be explained by Table 10 which shows the number of terms in the enlarged candidate dictionaries after each iteration. It is easy to understand that the decrease of the performance may be caused by adding too many terms which may contain noise. The stability of the performance when we run more iterations is due to the fact that the candidate dictionaries have saturated after the first three iterations. It deserves our further study to control the increase rate of the candidate dictionaries by tuning the probability thresholds so that we can prevent adding noise.

7. CONCLUSION

In this paper, we studied the problem of personal name classification in Web queries, which is fundamentally different from conventional named entity extraction and text classification due the sparse information contained in Web queries. We put forward an probabilistic algorithm based on the construction of probabilistic dictionaries and personal name grammars, which can generate the probability of a query being a personal name. Through this method, we can trade-off the precision and recall easily so that we can cater for different application scenarios. By comparing our proposed method with two kinds of baselines on some real-world data sets, we can see that our methods outperform the baselines by more than 11.3% in terms of F1. We also stud-

Table 10: The number of terms in the enlarged candidate dictionaries after each iteration

	0	1	2	3	4	5
#First-Name Term	4,896	584,576	669,046	701,563	701,564	701,564
#Last-Name Term	85,668	225,110	952,551	993,155	994,542	994,542

ied several key aspects of our methods including methods of probability estimation, automatic construction of golden dictionaries and candidate dictionaries.

To construct the probabilistic dictionaries, we hope to exploit more comprehensive term contexts. Besides that, we will also try more precise methods to define name term contexts by leveraging the existing named entity recognition algorithms. How to control the parameters when we enlarge golden dictionaries and candidate dictionaries is another direction of our future work.

8. REFERENCES

[1] E. Alpaydin. *Introduction to Machine Learning*. The MIT Press., 2004.

[2] J. Artiles, J. Gonzalo, and F. Verdejo. A testbed for people searching strategies in the www. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 569–570, New York, NY, USA, 2005. ACM Press.

[3] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416, New York, NY, USA, 2000. ACM Press.

[4] H.-H. Chen and G.-W. Bian. White page construction from web pages for finding people in internet. *International Journal of Computational Linguistics and Chinese Language Processing*, 3(1):75–100, 1998.

[5] Z. Chen, L. Wenyin, and F. Zhang. A new statistical approach to personal name extraction. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 67–74, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[6] H. L. Chieu and H. T. Ng. Named entity recognition with a maximum entropy approach. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 160–163, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[7] H. K. Dai, L. Zhao, Z. Nie, J.-R. Wen, L. Wang, and Y. Li. Detecting online commercial intention (oci). In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 829–837, 2006.

[8] C. Dozier. Assigning belief scores to names in queries. In *HLT '01: Proceedings of the first international conference on Human language technology research*, pages 1–5, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[9] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 168–171, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[10] W. Ho, A. Smailagic, D. P. Siewiorek, and C. Faloutsos. An adaptive two-phase approach to wifi location sensing. In *PerCom Workshops*, pages 452–456, 2006.

[11] T. Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *ECML*, pages 137–142, 1998.

[12] V. Krishnan and C. D. Manning. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *ACL-COLING'06: Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 2006.

[13] U. Lee, Z. Liu, and J. Cho. Automatic identification of user goals in web search. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 391–400, 2005.

[14] C. Manning and H. SchÍźtze. *Foundations of Statistical Natural Language Processing*. MIT Press., Cambridge, MA, 1999.

[15] F. Peng, D. Schuurmans, and S. Wang. Augmenting naive bayes classifiers with statistical language models. *Inf. Retr.*, 7(3-4):317–345, 2004.

[16] J. Platt. Probabilities for sv machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[17] D. Shen, R. Pan, J.-T. Sun, J. J. Pan, K. Wu, J. Yin, and Q. Yang. Query enrichment for web-query classification. *ACM Transaction on Information System.*, 24(3):320–352, 2006.

[18] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

[19] W. tau Yih, J. Goodman, and V. R. Carvalho. Finding advertising keywords on web pages. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 213–222, New York, NY, USA, 2006. ACM Press.

[20] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, second edition edition, 1979.

[21] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[22] X. Wan, J. Gao, M. Li, and B. Ding. Person resolution in person search results: Webhawk. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 163–170, New York, NY, USA, 2005. ACM Press.

[23] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang. Clustering user queries of a search engine. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 162–168, 2001.

[24] Y. Yang. An evaluation of statistical approaches to text categorization. *Inf. Retr.*, 1(1-2):69–90, 1999.