

Entity Identification on the Web

Charu Tiwari

Roll No. 03329029

Kanwal Rekhi School of Information Technology,
Indian Institute of Technology Bombay,
Powai, Mumbai-400 076. India

As part of the first stage presentation of M.Tech.Thesis Project
Under the guidance of **Prof. Sunita Sarawagi**

Abstract

Searching for information about a particular entity is the most frequent activity performed on the Web. Many a times, it is also needed to integrate information available from various online sources. The presence of multiple entities with the same name makes it difficult for a user to browse through the results. In such cases, it is often useful to be able to determine when two referents (pages) are referring to the same physical entity. Since the information available on the Web is in human readable form, it is comparatively easier for humans to make such a distinction by understanding, and navigating through links. But, since the Web hosts a massive amount of information, it is desirable to automate this process, and allow a machine to make such a distinction. In this project, we address this problem by finding suitable methods from the field of Information Extraction. The solution should be such that given the name of the entity the user is searching for, and a number of Web documents obtained as a result of the search query for that name, the machine should be able to cluster together all the documents referring to same physical entity. This report gives a description of the preliminary work done in this direction.

1 Introduction

The Web today hosts an astronomical amount of data. Moreover, this information is increasing day by day. Although, the information is available in human-readable form, it is very difficult and monotonous to manually perform certain tasks on the Web. For example, searching for job advertisements, segmenting citations to fill up a database, and identifying duplicates.

Searching for a particular entity is the most frequent activity on the Web. Another frequent Web related task is the integration of information collected

from various online sources to facilitate easy access of information to the users. Both these operations present the problem of duplicate identification (also referred to as deduplication, or disambiguation), and co-reference identification. Although, much work has been done to identify duplicates from among given structured documents such as database records, and semi-structured documents such as citations, and addresses, the disambiguation problem when applied to unstructured documents is still under-explored. In this project, we address a problem closely related to the problem of deduplication. In this project, we target towards developing an efficient method for solving the problem of “Co-reference Identification”. We attempt to develop a system, which when given the name of the entity the user is searching for, and a number of Web documents obtained as a result of the search query on that name, should be able to cluster together all the documents referring to same physical entity. The problem is challenging, as it is not trivial for a machine to determine the context of the document.

The remainder of this report is organised as follows: Section 2 gives the motivation for working towards the solution of this problem. Section 3 formally describes the problem of entity identification, highlighting some of the challenges that we are facing, followed by a brief survey of the related work done in this area, in Section 4. Finally, Section 5 gives some of the possible approaches that can be considered.

2 Motivation

A user very often searches for a particular entity on the Web. Usually, there are a number of entities with the same name. In order to narrow down the number of results obtained, the user has to supply additional keywords related to the particular entity he/she is looking for. This is undesirable as it may result in some of the valid pages being left out. So, it is desired to perform entity identification on the Web without much user intervention. Following are some of the motivating examples.

2.1 Information Integration

Today, a number of online resources host massive amount of useful information. Also, one resource might provide information about some entity that the other resource doesn't. Often, it is desired to integrate information gathered from such resources, and collect information belonging to the same entity. This requires identifying when two similar names actually refer to the same physical entity. Manually performing such task is tedious and time consuming. The question to be asked is: Can we automate this process of entity identification so as to facilitate the process of Information Integration?

2.2 Hunting for information about a specific entity

A user, while searching for information about some entity on the Web, may be presented with a number of documents, which might refer to different physical entities. This requires a user looking for a particular entity to manually read, understand, and then sort the documents to pick those that are relevant to him/her. E.g., a user queries a search engine for information on “Tom Mitchell”. The search engine will give a list of documents, not all referring to the “Tom Mitchell” the user is looking for. This is because more than one person can have the same name. Some result entries might refer to the “Tom Mitchell” who is a professor, others might relate to some musician, while still others might refer to an entirely different person. Now, a user searching for information on “Tom Mitchell” who is a professor, will have to read the resultant documents, and identify all those pages that are of relevance to him/her. It would be much convenient if this process can be automated. In this project, we try to find methods so that such an entity-identification can be done automatically, and the system presents the user with a number of clusters each referring to a separate entity.

3 Problem Statement

Starting from a given entity name, and a number of Web documents obtained as a result of a search query performed on the entity name, the problem is to cluster together all those documents referring to the same physical entity. The problem is challenging as the same name might refer to two entirely different physical entities.

3.1 Inputs

1. The name N of the entity being searched.
2. A list L of URLs (Uniform Resource Locators) obtained as a result of a Web search query on N .
3. Training examples with correct clustering of the documents.

The access to the Web through the URLs in the list L is assumed.

We also model the available Web as a graph $G(V, E)$ consisting of vertex set V and edge set E , where a vertex is a webpage and an edge $e = \langle u, v \rangle$ represents a hyperlink pointing from a webpage u to a webpage v .

3.2 Output

A clustering of the L documents, such that all co-referencing documents are in the same cluster, and the non-coreferencing documents are in different

clusters. The objective is to maximize the goodness of the clustering so obtained. One way of measuring the goodness of a given clustering is by considering the precision and recall values for each document in each cluster separately, and then computing the overall precision and recall values.

3.3 Challenges

Section 2 gives some of the examples describing the problem of entity identification. Here we describe a few more to understand the challenges involved in this problem.

- Suppose a user needs to search for some information about a school named “Carmel Convent” located in the city of “Bhopal”. Since there may be a number of schools with this name all over the world, with most of them having their information on the Web, the user will have to search through all the webpages given by a search engine to find the one of relevance to him/her. It is also possible that there may be more than one “Carmel Convent” school in “Bhopal”. The main challenge here is — how to gather information that will help us disambiguate homogeneous entities having the same name?
- Another case is to search for a person named “Flower”. Now, the result is likely to contain not only the documents referring to different persons with the name “Flower”, but also the documents pertaining to the floral world. So, another challenge is to be able to distinguish between heterogeneous entities. However, we believe that this problem should not be a difficult one.

So, the main challenges are:

- To be able to extract information from the input documents that will help distinguish between both homogeneous as well as heterogeneous entities having the same name.
- To be able to navigate through the Web and extract the desired information. Here, we also need to be able to determine when to stop the search.

4 Related Work

This section presents some of the work done in areas related to our problem. Closely related to our problem of entity identification is the problem of duplicate detection, and anti-aliasing.

4.1 String Distance Metrics

String matching plays an important role in the process of comparing two documents. Due to variation in the formats, typographical errors, and abbreviations, it becomes difficult to identify when two blocks of text represent the same information. [2], [5], and [4] describe some of the commonly used metrics for string comparisons, some of which are described in this section. [5] also gives description of a Java based tool, called “**SecondString**”, which provides most of the similarity metrics described here.

The various string metrics can be broadly classified as follows:

1. **Edit distance based metrics:** These metrics map a pair of strings s and t to a real number r , where a smaller value of r indicates greater similarity between s and t . **Edit distance** between two strings is defined as the cost of the best sequence of edit operations, namely substitution, insertion, and deletion, to convert one string to another. An example is the “Levenstein distance” which gives equal cost to all the three operations of substitution, insertion, and deletion. Affine edit distance functions give lower cost to a sequence of insertions and deletions. Another metric called “Jaro” similarity takes into account the number and order of common characters.
2. **Token based metrics:** These metrics consider given text as a bag of words. “Jaccard” similarity is simply the ratio of the number of common words between the two strings being compared, to the number of words in their union. One of the most frequently encountered metric in the field of Information Extraction is “TFIDF/cosine” similarity. This metric considers each string as a vector, where the vector dimensions give the weight of the corresponding word. TFIDF similarity between two strings is then proportional to the dot product of the vectors representing the two strings. More weight is given to a term appearing less frequently in the corpus, and vice-versa. Also, the weight of a term increases with its frequency in the current string. Mathematically,

$$\begin{aligned} TF - IDF(S, T) &= \sum_{w \in S \cap T} V(w, S) \cdot V(w, T) \\ V'(w, S) &= \log(TF_{w,S} + 1) \cdot \log(IDF_w) \\ V(w, S) &= V'(w, S) / \sqrt{\sum_{w'} V'(w', S)^2} \end{aligned}$$

where $TF_{w,s}$ is the frequency of word w in S , IDF_w is the inverse of the fraction of names in the corpus that contain w .

3. **Hybrid metrics:** These metrics are a combination of the string and token based metrics. An example is “Monge-Elkan” similarity which

breaks a long string into a sequence of substrings for the purpose of comparisons. E.g., two strings s and t are broken as $s = a_1 \dots a_K$ and $t = b_1 \dots b_L$. Then the similarity between s and t is defined as:

$$sim(s, t) = \frac{1}{K} \sum_{i=1}^{i=K} \max_{j=1}^{j=L} sim'(a_i, b_j)$$

where sim' is some secondary distance function like edit distance.

Another hybrid metric found very useful is the “Soft TFIDF metric”, which is similar to the TFIDF metric with the modification that it considers not only the common words in the two strings (as considered by TFIDF), but also similar words. Soft TFIDF measure allows for some errors like OCR errors, and typos, to occur, and still allows the strings to match. So, it performs better than TFIDF in most cases.

[4] divides the string metrics into two main categories, viz static similarity measures and learnable similarity measures, depending on whether or not they adapt to the domain to which they are applied. Static similarity measures, as the name suggests, do not change with the particular domain to which they are applied. Such metrics do not require any training. Contrast to this, learnable similarity metrics require training data so as to adapt to the particular domain under consideration. Consider matching of two addresses, which have been segmented into fields like name, street, city, country, and zip code. Often, the street field is found to use the abbreviation **St.** for **Street**; no such abbreviations are common in the country field. The static similarity metrics treat both these fields equally, failing to accurately estimate the distance. Hence, precise similarity computations require adapting the string similarity metrics for each field according to the domain.

4.2 Anti Aliasing on the Web

A problem related to entity identification is the identification of authors of the text given only the text and the alias using which the text was posted. The problem is challenging as the same author can use multiple aliases for posting; for reasons of privacy, as well as fun.

The main idea behind anti-aliasing is that each person has an inherent style of writing that remains the same despite of using a different alias. [7] suggests a method to the above mentioned problem of anti-aliasing. Given a number of aliases, and the text written by them, the idea is to represent each alias by a vector. Each dimension of the vector corresponds to a word, and the corresponding value for the dimension gives the probability of that word appearing in the text for the alias. [7] also suggests a number of similarity measures such as TFIDF similarity, and KL similarity. It was found that KL similarity gives better performance. For two aliases a and b , the KL

similarity measure is defined as:

$$D(a||b) = \sum_i p_{a_i} \frac{\log p_{a_i}}{\log p_{b_i}}$$

where p_a and p_b give the probability distribution for aliases a and b respectively. The KL similarity measures the number of extra bits required to encode the text of a using the optimum code for alias b .

After computing the similarity for each pair of aliases, for each alias, all other aliases are ranked according to their similarity values. This is done by sorting the aliases based on their similarity. Using this rank value measure, a clustering algorithm, which merges those clusters such that cohesion is minimised, is used to cluster together the aliases belonging to the same author. The idea behind the minimization of cohesion is to ensure that all the aliases included in one cluster should be ranked high in the list for each other. F-measure is used to measure the goodness of the clustering so obtained. It was found that with the increase in the number of messages available per alias, the F-Measure increased. Also, it was found that the other features such as emoticons, and misspellings, do not contribute much in increasing the F-Measure. One problem with this method is that it makes use of just the frequency of words to compute the similarity between two aliases. Since, people discussing on the same topic tend to use same vocabulary during the discussion, this method tend to cluster together aliases belonging to different people, just because they engage heavily in the discussion about the same topic. This method can be improved by taking into account the capitalisation of words, references to people and places used by the authors, and signatures.

4.3 Active Learning Led Interactive Alias Suppression

Consider the problem of integrating citation entries, or addresses from two different sources. It is expected that some of the entries will be duplicates. So as not to replicate information, duplicate elimination becomes important. Earlier work in this area made use of just the number of common words between the entries to determine the similarity between them. But, many a times two different entries do contain a number of common words. Also, because of the use of abbreviations, and difference in style of writing, counting just the number of common words is not of much use. [8] suggests an efficient method for suppressing aliases using Active Learning.

Active Learning is a semi-supervised learning process that takes only a small amount of labelled data, and huge amounts of unlabelled data as input. An initial model is learnt using the labeled training data. The model is then applied to predict the class labels of the unlabeled instances. The uncertainty involved in the prediction is used to select the tuple which is presented to the user to label it. The classifier is then trained again after including the newly labelled data for training. This method continues till

sufficient amount of accuracy is achieved. This method greatly reduces the amount of labelled data required to train the model.

The deduplication algorithm suggested in [8] takes as input a number of similarity functions [2] [4], a small amount of labelled training data, and a huge amount of unlabelled data. Assuming that the data has already been segmented into the required fields (such as name, street, city, and zip code, in case of addresses), the idea is to form all possible pairs of records (citations, addresses, and the like). Using the similarity functions input by the user, each pair of records is represented by a new record, the attributes of which are the values of the similarity functions, as computed for the pair. The similarity function is applied according to the type of field. E.g., edit distance like functions for the text fields, and absolute difference for the integer fields. Then, a classifier is learnt using the idea of active learning. For the purposes of active learning, both the uncertainty as well as the representativeness of an instance is taken into account. [8] also suggests a number of methods for computing the uncertainty of an instance. Most frequently, a committee of classifiers is formed, and the disagreement among the classifiers gives a measure of the uncertainty of the instance. Most uncertain instance is usually the one which is presented to the user for labelling. The idea is that the most uncertain instance is the one whose correct label will help the classifier learn most. But, to prevent the outliers from affecting the model, the instance selected should be representative of the underlying data. When a huge amount of data is submitted to the system for duplicate detection, the number of pairs generated will be enormous, and computations will be very expensive. [8] suggests some optimisations for efficient evaluation of the predicate learnt by the classifier. Some of them are as follows:

- Rather than taking the cross product of the records and then computing the predicate, to incrementally evaluate the predicate by using grouped evaluation.
- To evaluate simpler predicates before the expensive ones.
- To precede expensive predicates by canopies which filter the data.

4.4 Syntactic Clustering of the Web

Surfing on the Web is the most common usage of the Internet today. It would be very useful if we can present related documents to a user searching for information on a particular topic. To facilitate this, it would be required to cluster together similar documents so that they can be retrieved efficiently.

[3] discusses a possible approach to identification and clustering of similar documents. Resemblance and containment are the two measures that are considered for determining similar documents. The idea is to first convert each document into a set of shingles, where a shingle is defined as a sequence

of tokens. Each document is then represented by a sketch, which consists of sets F and/or V . A random permutation is performed on the set of singles for the document, and the first s shingles are selected to give the F set for the document. The V set is obtained by assigning numbers to the set of shingles for the document, and then taking the set of all those elements that are $0 \bmod m$, where m is some suitable number. The F set is used in determining the similarity, while V set is useful in determining containment of one document in another. The sketch for each document is then expanded into a file containing $\langle shingle, docID \rangle$ pairs. Using these files a new file containing $\langle doc1ID, doc2ID, count \rangle$ is generated, where ‘count’ gives the number of common shingles between the two documents with ids $doc1ID$, and $doc2ID$. Next, for each document pair, resemblance and containment values are computed using these count values. Those pairs for which the resemblance value exceeds a predefined threshold are considered to be similar to each other. Using a clustering algorithm such similar documents are clustered together. Finally, we get a number of clusters each containing syntactically similar documents. Another idea that is proposed is “Super Shingling”, where a super shingle is a sequence of shingles. If two documents agree on atleast one super shingle, then the documents are considered to be similar, and will be clustered together. But, the disadvantage of this approach is that it cannot be applied to small documents which contain less data.

Syntactic clustering of the Web has a number of applications. E.g., by giving a single url, a user will be able to find similar documents from the Web; when a url becomes dead then the user will be able to get the similar document; and identification of illegal plagiarism.

4.5 Entity-Based Cross-Document Coreferencing Using the Vector Space Model

Some of the earlier work done in the area of entity identification on the Web is discussed in [1]. [1] suggests a method to identify when two documents refer to the same entity. The paper proposes a method for cross-document reference resolution. Given a number of documents as input, the method results in a number of clusters/chains, such that all similar documents are chained together.

First each document is submitted to a system (called the CAMP system), which identifies all the coreference chains in the document. Each coreference chain corresponds to an entity. The entity chain of interest is selected, and the document is submitted to a summary extractor which extracts the summary for that entity. Now, each such summary represents an entity. The goal is to group together summaries that belong to the same entity. For this, each summary is represented by a vector of terms. Using the TFIDF similarity measure [2], similarity between pairs of documents is computed.

If similarity comes out to be more than a predetermined threshold value, the summaries are assumed to represent the same entity. This way the documents are chained together so that all the documents in a chain correspond to the same entity. Transitive relation is assumed while chaining the documents together. [1] also gives an evaluation method to compute the goodness of chains so formed. Rather than taking into account just the number of missing links for evaluation, a more intuitive method is to compute the recall and precision values for each entity in the chain separately. The total precision/recall of each chain is then equal to a weighted sum of the precision/recall values for all the entities within the chain. This method is called the “B-cubed method”.

The weakness of this method is that it takes into account only the frequency of common words for computing the coreference similarity. Thus, two documents that refer to different entities, but have a number of words in common may be chained together.

4.6 Disambiguating People in Search

[6] gives some of the initial work that has been done in the area of disambiguating people on the Web. It has been observed that webpages cannot assume context. Whenever a person is referred to by a webpage, some information is also associated with the name so that the reader can determine the identity of the person being referred to. Such information includes location, email, profession, and notable achievements. If this information is available on each webpage displayed as a result of the search, then it can be utilised to identify the person the user is looking for. This fact is made use of in the method suggested in [6].

Consider the example of a user searching for a person on the Web. The user will be presented with a number of result pages. The algorithm then asks the user to select the page describing the person he/she is looking for. This page is referred to as the base page. The idea is to represent each page of the result by a sketch, where a sketch is a vector, the attributes of which are the values for the functions like first name, last name, profession, and location. A sketch can also be imagined to be a graph with all the edges sharing one end point representing the entity under consideration; other end points giving the values of the attributes for the page. Since these attribute values may have been extracted using some statistical methods, a probability can be associated with each edge. Weights can also be assigned to the edges depending on their importance in disambiguating people. These weights can either be learnt or obtained through good statistical data. Also, the method makes use of prior knowledge (like using databases) about the domain to determine the edge weights depending on the value of the corresponding attribute. E.g., given the fact that the two people (referred to by two different pages) work in the same company, the probability of their being referring to

the same person will be more if the company is small. So, more weight can be assigned to the corresponding edge. The method compares the remaining pages with the sketch of the base page. The result is the reordered list with the relevant pages ranked high.

The weakness of this method is that it assumes the presence of some existing database for the domain under consideration. Also, different pages use different descriptions, which only partially overlap. Another extension that is possible is to be able to compare pages referring to different types of entities but having the same name. E.g., the term “Matrix” can refer to the mathematical concept as well as film.

4.7 Relevance

We find that the work done in [1] and [6] is most closely related to our problem. However, the methods discussed in these cannot be applied directly as they make assumptions that may not be valid in many cases. [6] highlights the fact that, in a webpage, there is always some amount of information available in the vicinity of the required entity. This idea can be very useful in entity identification. But, it assumes the existence of an external database to help identify relevant information, and assign appropriate weights to different features. Availability of such databases cannot always be assumed. [1] describes a way to identify co-references, but it assumes the availability of similar information in all the co-referring documents. This assumption does not hold on the Web. Two documents may have entirely different contents, but may still refer to the same entity. In such cases, the idea (proposed by us) of navigating through the Web to gather relevant information, will be of help.

[7] gives a way to demystify authors’ identity. But, it does not relate to our problem directly. It makes use of the writing styles of the authors to disambiguate their identities, while we need to work on the webpages, and not all the webpages that refer to a particular entity are written by the entity itself.

The idea of “Active Learning” used in [8] needs to be explored for its application to our problem, as it will help reduce the amount of labelled training data required. The paper suggests a way to deduplicate semi-structured records like bibliographic entries. So, the method cannot be applied in our case directly. However, the ways suggested to identify same yet different appearing strings will be very helpful.

[3] presents a new idea of using shingles to identify similar documents. Also, the idea of permuting the words and then picking some of them is novel. The idea of making use of shingles can be explored for its application to our problem.

The similarity measures described in [2], [5], and [4] will prove to be helpful in the identification of similar strings. Also, the Soft-TFIDF measure

is expected to be of much use in the disambiguation of heterogeneous entities.

5 Work Done

5.1 Literature Survey

Section 4 gives the literature survey done to understand the problem, and the work that has already been done in this area. Next, we give some of the possible solutions.

5.2 Probable Solutions

In this section, we propose a probable, naïve solution to our problem. Two cases need to be considered:

5.2.1 Distinguishing between heterogeneous entities

We need to be able to distinguish between two documents each referring to entities having the same name but belonging to entirely different domains. E.g., “Matrix” can refer to the mathematical concept as well as the film “Matrix”. We believe that the TFIDF similarity measure should be good enough to distinguish between heterogeneous entities having same name. This is so because the documents referring to these entities are likely to contain words from entirely different domains. E.g., in the case of two documents each referring to “Matrix”, the one referring to the “Matrix” as a mathematical concept is likely to contain more words related to mathematics, while that referring to the movie is likely to contain more words related to entertainment and media.

5.2.2 Distinguishing between homogeneous entities

We also need to be able to distinguish between entities of the same type sharing the same name. This is more challenging, as the documents referring to both will contain words from similar domain. So, TFIDF measure is not expected to give the desired results.

Whenever a name is referred to by a webpage, some information related to it is usually present in the locality of reference. E.g., when searched for a school named “Carmel Convent”, the resulting documents contained the location information (like city, state, and country) in the locality of reference. If we can identify such information, and can give more weights to the relevant words appearing near the entity name, the performance is expected to improve. Also, the possibility of utilising the URL information contained in the documents to facilitate entity identification, can be explored. Since, we assume access to the Web through the links in the documents, these outlinks can also be explored to get a better idea about the identity

being referred to from the main page. An important issue is — how deep to explore while hunting for information that can disambiguate the entities under consideration? However, these are just proposed solutions, and their effectiveness still need to be explored.

References

- [1] Amit Bagga and Breck Baldwin. Entity-Based Cross-Document Coreferencing Using the Vector Space Model. In *COLING-ACL*, 1998.
- [2] Mikael Bilenko, Ray Mooney, William Cohen, Pradeep Ravikumar, and Steve Fienberg. Adaptive Name Matching in Information Integration. *IEEE Computer Society*, 2003.
- [3] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. In *Proc. Sixth Int'l. World Wide Web Conference*, pages 391–404. WWW Consortium, 1997.
- [4] William Cohen, Pradeep Ravikumar, and Steve Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. *IIWeb Workshop*, 2003.
- [5] William Cohen, Pradeep Ravikumar, and Steve Fienberg. A Comparison of String Metrics for Matching Names and Records. *KDD-2003 Workshop on Data Cleaning and Object Consolidation*, 2003.
- [6] R. Guha and A. Garg. Disambiguating people in search. *ACM*, 2004.
- [7] Jasmine Novak, Prabhakar Raghvan, and Andrew Tomkins. Anti-Aliasing on the Web. *ACM*, 2004.
- [8] Sunita Sarawagi and Anuradha Bhamidipaty. Interactive Deduplication using Active Learning. *SIGKDD 02 Edmonton, Alberta, Canada*, 2002.
- [9] Sunita Sarawagi and Alok Kirpal. Scaling up the ALIAS Duplicate Elimination System: A Demonstration. *SIGKDD 02 Edmonton, Alberta, Canada*, 2002.