ISSN 2348 – 9928
IJAICT Volume 3, Issue 10, October 2016 Doi:01.0401/ijaict.2014.07.06 Published on 05 (10) 2016

ANNOTATING THE DOCUMENT USING CONTENT AND QUERYING VALUE

Dr S. Ambareesh
Associate Professor,
Computer Science and Engineering,
Vemana Institute of Technology,
Bangalore, India

Ms. Thejaswani T Rao
Bachelor of Engineering,
Computer Science and Engineering,
Vemana Institute of Technology,
Bangalore, India

Ms. D. C. Jayalakshmi
Bachelor of Engineering,
Computer Science and Engineering,
Vemana Institute of Technology,
Bangalore, India

Abstract— There are many application domains where users create and share information. Extraction algorithms facilitate the extraction of structured relations, which are often expensive and inaccurate, especially when operating on top of text that does not contain any instances of the targeted structured information. We present a novel alternative approach that facilitates the generation of the structured metadata by identifying documents that are likely to contain information of interest and this information is going to be subsequently useful for querying the database. Our approach relies on the idea that humans are more likely to add the necessary metadata during creation time. As a major contribution of this paper, we present algorithms that identify structured attributes that are likely to appear within the document, by jointly utilizing the content of the text and the query workload.

Keywords— Attributes Suggestion, Annotation, Datasets.

I. INTRODUCTION

Current information sharing tools, like content management software (e.g., Microsoft SharePoint), allow users to share documents and annotate (tag) them in an ad-hoc way. Similarly, Google [1] allows users to define attributes for their objects or choose from predefined templates. This annotation process can facilitate subsequent information discovery. Many annotation systems allow only "untyped" keyword annotation: for instance, a user may annotate a weather report using a tag such as "*Storm Category 3*".

Annotation strategies that use attribute-value pairs are generally more expressive, as they can contain more information than untyped approaches. Many systems, though, do not even have the basic "attribute-value" annotation that would make a "pay-as-you go" querying feasible. Annotations that use "attribute-value" pairs require users to be more principled in their annotation efforts. Difficulties results in very basic annotations, that is often limited to simple keywords. Such simple annotations make the analysis and querying of the data cumbersome. Users are often limited to plain keyword searches, or have access to very basic annotation fields, such as "*creation date*" and "*owner of document*".

The main goal of CADS is to lower the cost of creating annotated documents that can be immediately used for commonly issued semi-structured queries. Our key goal is to encourage the annotation of the documents at creation time, while the creator is still in the "document generation" phase, even though the techniques can also be used for post generation document annotation. Once uploaded, CADS analyzes the text and creates an adaptive insertion form. The form contains the best attribute names given the document text and the information need, and the most probable attribute values given the document text. The creator can inspect the form, modify the generated metadata as- necessary, and submit the annotated document for storage.

II. FRAMEWORK AND PROBLEM DEFINITION

In this section, we present the notation that we use in the rest of the paper and describe the problem setting. As discussed in Section 1, our goal is to suggest annotations for a document. We define a document d as a pair (d_t, d_a), composed of the textual content d_t and the set of existing user annotations da. We use to denote the complete and optimal set of annotations for d. The d^{opt}_a serves as a conceptual baseline, i.e., is created by an oracle with perfect knowledge of the domain of d (e.g., disaster management) and, of course, d^{opt}_a is unknown to the algorithm that is trying to estimate as accurately as possible the d^{opt}_a .

Table 1 summarizes the notation presented. Using the above, we define our problem. A straightforward goal is to produce and display in the adaptive insertion form d^{opt}_a a , given d_t; this is usually a very large set of annotations. Even if we could produce all relevant annotations, a large number of such

ISSN 2348 – 9928
Doi:01.0401/ijaict.2014.07.06 Published on 05 (10) 2016

annotations may also overwhelm the user who must examine, modify, and approve all the suggestions. Hence, our efforts focus not only on identifying the potential annotations fields that exist in d^{opt}_a, but also to *rank* them and display on top the most important ones. Since the goal of annotations is to facilitate future querying, we want the annotation effort to focus on generating annotations useful for the queries in the query workload W. So, if users are willing to fill-in at most k annotations for a single document (where k is arbitrary, but fixed), our goal is to generate a subset of d^{opt}_a while under the constraint of at-most-k- annoations. This set of annotations should be the one that increases the visibility of document d in W, that is, maximizes the number of queries that retrieve d.

Table 1: Notation

A	Attributes used in the union of W and $\mathcal{D}$
A_j	Attribute in A
d	Document
d_t	Document text for d
d_a	Document annotations for d
$\mathcal{D}$	Repository
k	Maximum number of suggestions
$Q = q_1, q_2 \dots q_m$	Query
d_a^{opt}	complete and optimal annotations for d
W	Workload
$annotated(d, A_j)$	Document d is annotated with A_j
$use(A_j, q)$	Query q uses A_j
$\mathcal{P}$	System Prior
w	term
$score(A_j)$	Ranking function
D	Database
D_{A_j}	Database Documents annotated with A_j
$D_{A_j,w}$	Database Documents annotated with A_j that contains term w
β_i	Coefficients for Bernoulli Model

III. ATTRIBUTES SUGGESTION

In this section we study and propose solutions for the "*attributes suggestion*" problem. From the problem definition we identify two, potentially conflicting, and properties for identifying and suggesting attributes for a document d:

- First, the attributes must have high *querying value* with respect to the query workload W. That is, they must appear in many queries in W, since the frequent Attributes in W have a greater potential to improve the visibility of d.

- Second, the attributes must have high *content value* with respect to d_t. That is, they must be relevant to d_t. Otherwise, the user will probably dismiss the suggestions and d will not be properly annotated.

We combine both objectives, in a principled way, using a probabilistic approach. Our theoretical model is similar to the idea of language models [2], with one key difference: our model assume that attributes are generated by *two* processes, in parallel: (a) By inspecting the content of the document and extracting a set of attributes related to the content of the document, following a probability distribution given by an (unknown to us) joint probability distribution $p(d_a, d_t)$; and (b) By knowing the types of queries that users typically issue to the database, following again an (unknown to us) joint probability distribution $p(d_a, W)$.

3.1 Conditional Independence given A_j and A_j

We denote with $p(A_j|W, d_t, \mathcal{P})$ be the posterior probability that document d is annotated with A_j , given the forecast of W,d and a prior belief P of CADS about the probability of adding A_j in any document.5 We define the score of attribute A_j as the odds that the attribute should appear in d_a. Using the Bayes theorem:

$$Score(A_j) = \frac{p(A_j|\mathcal{P}, W, d_t)}{p(\overline{A_j}|\mathcal{P}, W, d_t)} = \frac{p(\mathcal{P}, W, d_t|A_j) \cdot p(A_j)}{p(\mathcal{P}, W, d_t|\overline{A_j}) \cdot p(\overline{A_j})}$$

The numerator and denominator are equivalent to the joint distributions $p(P,W, d_t,A_j)$ and $p(P,W, d_t,A_j)$, respectively. Using the chain rule on both terms:

$$Score(A_j) = \frac{p(\mathcal{P}) \cdot p(A_j|\mathcal{P}) \cdot p(W|A_j, \mathcal{P}) \cdot p(d_t|A_j, \mathcal{P}, W)}{p(\mathcal{P}) \cdot p(\overline{A_j}|\mathcal{P}) \cdot p(W|\overline{A_j}, \mathcal{P}) \cdot p(d_t|\overline{A_j}, \mathcal{P}, W)}$$

If W is independent of P, given A, and d_t is independent of W,P, we simplify:

$$Score(A_j) = \frac{p(A_j|\mathcal{P}) \cdot p(W|A_j) \cdot p(d_t|A_j)}{p(\overline{A_j}|\mathcal{P}) \cdot p(W|\overline{A_j}) \cdot p(d_t|\overline{A_j})}$$

Our prior belief P is independent of $p(A_j)$, as we are not using any external knowledge to affect the estimates. So, the above equation can be further simplified to:

$$Score(A_j) = \frac{p(A_j|W)}{1 - p(A_j|W)} \cdot \frac{p(d_t|A_j)}{p(d_t|\overline{A_j})} \qquad (1)$$

Equation 1 is our score function. The first term represents the likelihood of producing A_j , given the workload W. We refer to that term as *querying value* as it expresses the "relevance" of the attribute to the query workload. The second term, which

ISSN 2348 – 9928
Doi:01.0401/ijaict.2014.07.06 Published on 05 (10) 2016

we refer to as *content value* is the likelihood of observing the content d_t given that the attribute A_j appears in the document.

3.2 Estimation Process

We now present our process for estimating the values of the parameters in Equation 1.

Querying Value: Let WA_j = {Q 2 W: use (Q,A_j)} be the set of queries in W that use A_j as one of the predicate conditions. We use Laplace smoothing [3] to avoid zero probabilities for the attributes that do not appear in the workload, we have:

$$p(A_j|W) = \frac{|W_{A_j}| + 1}{|W| + 1} \qquad (2)$$

in d_t, which is a typical assumption when dealing with textual data (e.g., in probabilistic information retrieval, text classification, language models, etc.) We have:

$$p(d_t|A_j) = \prod_{w \in d_t} p(w|A_j) \qquad (3)$$

where the product goes over all terms w in d_t.

Let DA_j = {d 2 D : annotated(d,A_j)} be the set of documents in the database D, annotated with the attribute A_j .

Let DA_j ,w = {d 2 D : annotated(d,A_j) ^ contains(d_t,w)} be the set of documents in the database that are annotated with A_j and also contain the word w in their text d_t. We estimate the probability of each term in Equation 3 as:

$$p(w|A_j) = \frac{|D_{A_j,w}| + 1}{|D_{A_j}| + |D| + 1} \qquad (4)$$

Again we use smoothing to avoid zero probabilities. For each term, the prior is uniform and we update the probability using the observed co-occurrences of A_j and w. In a similar way we define:

$$p(w|\overline{A_j}) = \frac{|D_{\overline{A_j},w}| + 1}{|D_{\overline{A_j}}| + |D| + 1} \qquad (5)$$

where we examine only documents that have been annotated and the attribute Aj was not added.

3.2 Conditional Independence among Forecaster Evidence

The second model is based on the assumption that each of the two forecasters has independent information about the event "attribute A appears in the document," (which is different than conditioning on A_j). We capture this information as a variable with distribution p that models the occurrence of the event "attribute A j appears in the document" as a Bernoulli experiment.

$$p_W = p(A_j|W), \text{ and } p_d = p(A_j|d_t)$$

Our final estimates is computed based on independent pieces of evidence, using the following model [4]:

$$p(A_j|W, d_t, \mathcal{P}) = \beta_0 \cdot \mathcal{P} + \beta_1 \cdot p_W + \beta_2 \cdot p_d \qquad (6)$$

IV. EFFICIENCY ISSUES AND SOLUTIONS

In this section, we discuss the algorithmic approaches that allow us to implement efficiently the algorithms described in the previous section. In particular, we show how pipelined algorithms can be employed [3] to compute the top-k attributes with the highest scores, where scores are defined using Equation 1 (Bayes strategy).

We observe that in both strategies the score is a monotonically increasing function (f(QV, CV) = CV ·QV for Bayes.

V. EXPERIMENTS

5.1 Datasets

Documents: For our experiments we use two document collections:

• The Emergency corpus consists of 270 documents, generated by the Miami-Dade Emergency Management Office. The documents are advisory, progress and situation reports submitted by various county stakeholders during the five days before and after Hurricane Wilma, which hit Miami-Dade County in October 2005.

• The CNET corpus consists of 4,840 electronic product reviews obtained from CNET. The dataset contains different kinds of products like cameras, video games, television, audio sets, and alarm clocks.

5.2 Annotations:

We generated annotations for the datasets, which we use as training and test data, to train and evaluate our algorithms.

To annotate the CNET reviews we used the CNET specifications page for each product. The page contains structured data for a product in the form of "attribute name value". Given that we are only interested in annotations that come from the document text (i.e. the product's review), we removed annotations that are not mentioned in any sentence in the review text. To decide when a sentence s is related

ISSN 2348 – 9928
Doi:01.0401/ijaict.2014.07.06 Published on 05 (10) 2016

(mentions) to an annotation $\mathcal{A} = (A_j, V_i)$, we used the containment ratio heuristic; specifically, we computed:

$$cr(\mathcal{A}, s) = \delta \frac{|A_j \cap s|}{|A_j|} + (1 - \delta) \frac{|V_i \cap s|}{|V_i|} \qquad (11)$$

5.3 Queries:

When generating the query workload for our datasets we had to address two main challenges. First, we did not have a query workload that was used to query the data sets in our disposal. So, we had to generate a workload, with an attribute distribution representing the user interests in a realistic way. Second, we had to create queries of the form attribute-value as described in Section 2.

Then, used the relative frequencies of the queries in the Google Insights/Trends to weight appropriately the workload in the results. For example, for the emergency data set we could see more queries related with the status of the schools on the city compared to queries asking for the status of the ports.

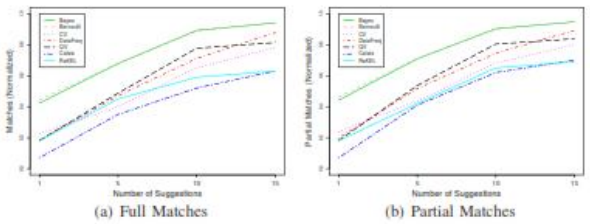
5.4 Attributes Suggestion Problem

In this experiment, we examine how the different strategies solve the Attributes Suggestion Problem, which is the core focus of our work. That is, if a strategy is used for attributes suggestion, how well [4] are the queries of the workload answered? To measure this we use the sum of documents returned by the queries in the workload, where a document is counted multiple times, once for every query that returns it. We refer to this measure as Full Match. We also consider a simpler variant, Partial Match, where we count how many query conditions are satisfied by the documents, that is, we view each query condition as a separate query. We first introduce the optimal suggestion techniques, which will be used as baselines to evaluate the strategies.

• ***OPTFullMatch*** suggests the subset of the ground-truth attributes for each document that maximize its query visibility in the query workload, that is, that satisfies the maximum number of queries. Miah et al. [5] prove that this problem is NP-Hard. However, given the relatively small size of our query workload, we were able to compute an exact solution using the exact algorithm from, following a brute-force approach, which took a significant amount of time but allowed us to measure exactly how close to the optimal each algorithm is.

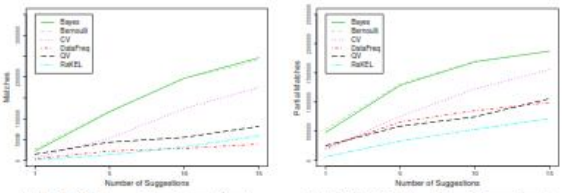
• ***OPTPartialMatch*** suggests a subset of the ground-truth attributes that maximize the number of query conditions satisfied. This can be computed making a single pass on the workload

We report the average coverage across all documents, where the coverage for one document is defined as the number of matches (or partial matches) divided by the number of matches of OPTFullMatch (or OPTPartialMatch, respectively). Fig 1 and 2 show the average coverage for full and partial matches for the strategies.



(a) Full Matches (b) Partial Matches

Fig 1 : Number of Full and Partial Matches in Emergency Dataset



(a) Full Matches (Non-Normalized) (b) Partial Matches (Non-Normalized)

Fig 2 : Number of Full and Partial Matches in CNET Dataset

VI. RELATED WORK

Collaborative Annotation: There are several system the favor the collaborative annotation of objects and use previous annotations or tags to annotate new objects. There has been a significant amount of work in predicting the tags for documents or other resources. Dataspaces and pay-as-you-go integration: The integration model of CADS is similar to that of dataspaces, where a loosely integration model is proposed for heterogeneous sources. The basic difference is that dataspaces integrate existing annotations for data sources, in order to answer queries. Our work suggests the appropriate annotation during insertion time, and also takes into consideration the query workload to identify the most promising attributes to add. Another related data model is that of Google Base [1], where users can specify their own attribute/value pairs, in addition to the ones proposed by the system. However, the proposed attributes in Google Base are hard-coded for each item category (e.g., real estate property). In CADS, the goal is to learn what attributes to suggest. Pay-as-you go integration techniques like Pay Go and [2] are

ISSN 2348 – 9928
Doi:01.0401/ijaict.2014.07.06 Published on 05 (10) 2016

useful to suggest candidate matchings at query time. However, no previous work considers this problem at insertion time, as in CADS. The work on Peer Data Management Systems is a precursor of the above projects.

VII CONCLUSION

We proposed adaptive techniques to suggest relevant attributes to annotate a document, while trying to satisfy the user querying needs. Our solution is based on a probabilistic framework that considers the evidence in the document content and the query workload. We present two ways to combine these two pieces of evidence, content value and querying value: a model that considers both components conditionally independent and a linear weighted model. Experiments shows that using our techniques, we can suggest attributes that improve the visibility of the documents with respect to the query workload by up to 50%. That is, we show that using the query workload can greatly improve the annotation process and increase the utility of shared data.

References

[1] Google,"Google base, http://www.google.com/base," 2012.

[2] S. R. Jeffery, M. J. Franklin, and A. Y. Halevy, "Pay-as-you-go user feedback for dataspace systems," in ACM SIGMOD, 2009.

[3] J. M. Ponte and W. B. Croft, "A language modeling approach to information retrieval," ACM SIGIR conference on Research and development in information retrieval.

[4] C. D. Manning, P. Raghavan, and H. Schutze,Available: http:// www.amazon.com / exec/obidos/

[5] M. Miah, G. Das, V. Hristidis, and H. Mannila, "Standing out in a crowd: Selecting attributes for maximum visibility," ICDE, 2008.