Assisting Document Annotation based on Content and Querying Value

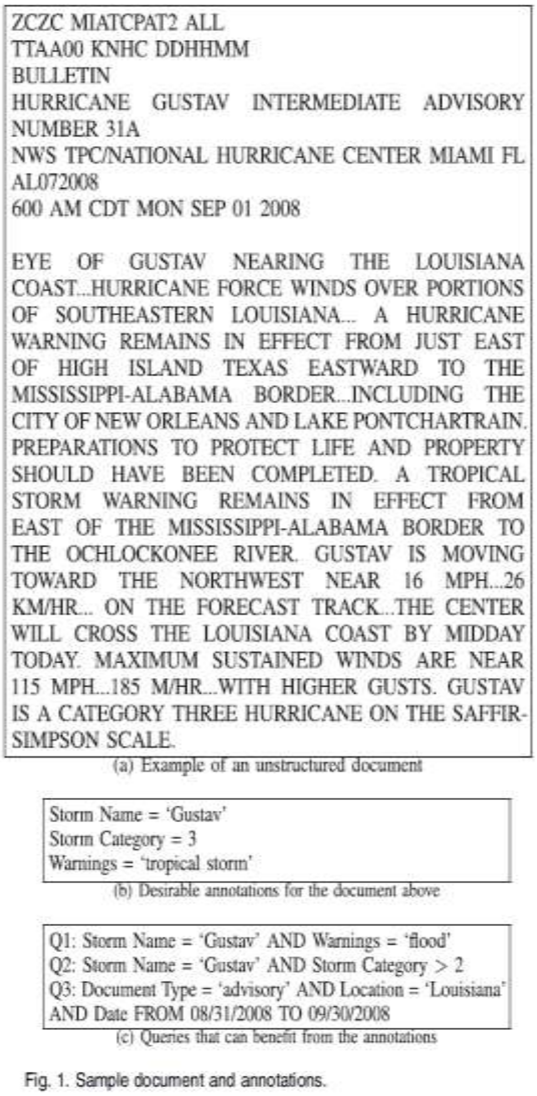
Pasupuleti Maneesha[1], Sayeed Yasin[2]
[1]M.Tech (CSE), Nimra College of Engineering & Technology, A.P., India.
[2]Head of the Department, Dept. of Computer Science & Engineering, Nimra College of Engineering & Technology, A.P., India.

Abstract — A annotation is metadata (e.g. a comment, explanation, presentational markup) attached to text, image, or other data. A large number of organizations today generate and share textual descriptions of their products, services, and actions. Such collections of textual data contain significant amount of structured information, which remains buried in the unstructured text. While information extraction algorithms facilitate the extraction of structured relations, they 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, if prompted by the interface; or that it is much easier for humans (and/or algorithms) to identify the metadata when such information actually exists in the document, instead of naively prompting users to fill in forms with information that is not available in the document. 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. Our experimental evaluation shows that our approach generates superior results compared to approaches that rely only on the textual content or only on the query workload, to identify attributes of interest.

Keywords — Document annotation, adaptive forms, and collaborative platforms

I. Introduction

The Open Annotation Core Data Model specifies an interoperable framework for creating associations between related resources, annotations, using a methodology that conforms to the Architecture of the World Wide Web. There are many application domains where users create and share information; for instance, news blogs, scientific networks, social networking groups, or disaster management networks. Current information sharing tools, like content management software (e.g., Microsoft Share- Point), allow users to share documents and annotate (tag) them in an ad hoc way. Similarly, Google Base [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. In such settings, the above information can be entered as (Storm Category, 3). A recent line of work toward using more expressive queries that leverage such annotations, is the "pay-as-you-go" querying strategy in Dataspaces [2]: In Dataspaces, users provide data integration hints at query time. The assumption in such systems is that the data sources already contain structured information and the problem is to match the query attributes with the source attributes. Many systems, though, do not even have the basic "attribute-value" annotation that would make a "pay-asyou- go" querying feasible. Annotations that use "attributevalue" pairs require users to be more principled in their annotation efforts. Users should know the underlying schema and field types to use; they should also know when to use each of these fields. With schemas that often have tens or even hundreds of available fields to fill, this task become complicated and cumbersome. This results in data entry users ignoring such annotation capabilities. Even if the system allows users to arbitrarily annotate the data with such attribute-value pairs, the users are often unwilling to perform this task: The task not only requires considerable effort but it also has unclear usefulness for subsequent searches in the future: who is going to use an arbitrary, undefined in a common schema, attribute type for future searches? But even when using a predetermined schema, when there are tens of potential fields that can be used, which of these fields are going to be useful for searching the database in the future? Such difficulties results in very basic annotations, if any at all, 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.” In this paper, we propose Collaborative Adaptive Data Sharing platform (CADS), which is an “annotate-as-youcreate” infrastructure that facilitates fielded data annotation. A key contribution of our system is the direct use of the query workload to direct the annotation process, in addition to examining the content of the document.



ZCZC MIATCPAT2 ALL
TTAA00 KNHC DDHHMM
BULLETIN
HURRICANE GUSTAV INTERMEDIATE ADVISORY NUMBER 31A
NWS TPC/NATIONAL HURRICANE CENTER MIAMI FL
AL072008
600 AM CDT MON SEP 01 2008

EYE OF GUSTAV NEARING THE LOUISIANA COAST...HURRICANE FORCE WINDS OVER PORTIONS OF SOUTHEASTERN LOUISIANA... A HURRICANE WARNING REMAINS IN EFFECT FROM JUST EAST OF HIGH ISLAND TEXAS EASTWARD TO THE MISSISSIPPI-ALABAMA BORDER...INCLUDING THE CITY OF NEW ORLEANS AND LAKE PONTCHARTRAIN. PREPARATIONS TO PROTECT LIFE AND PROPERTY SHOULD HAVE BEEN COMPLETED. A TROPICAL STORM WARNING REMAINS IN EFFECT FROM EAST OF THE MISSISSIPPI-ALABAMA BORDER TO THE OCHLOCKONEE RIVER. GUSTAV IS MOVING TOWARD THE NORTHWEST NEAR 16 MPH...26 KM/HR... ON THE FORECAST TRACK...THE CENTER WILL CROSS THE LOUISIANA COAST BY MIDDAY TODAY. MAXIMUM SUSTAINED WINDS ARE NEAR 115 MPH...185 M/HR...WITH HIGHER GUSTS. GUSTAV IS A CATEGORY THREE HURRICANE ON THE SAFFIR-SIMPSON SCALE.

(a) Example of an unstructured document

Storm Name = ‘Gustav’
Storm Category = 3
Warnings = ‘tropical storm’

(b) Desirable annotations for the document above

Q1: Storm Name = ‘Gustav’ AND Warnings = ‘flood’
Q2: Storm Name = ‘Gustav’ AND Storm Category > 2
Q3: Document Type = ‘advisory’ AND Location = ‘Louisiana’ AND Date FROM 08/31/2008 TO 09/30/2008

(c) Queries that can benefit from the annotations

Fig. 1. Sample document and annotations.

Example 1. Our motivating scenario is a disaster management situation, inspired by the experience in building a Business Continuity Information Network [3] for disaster situations in South Florida. During disasters, we have many users and organizations publishing and consuming information. For example, in a hurricane situation, local government agencies report shelter locations, damages in structures, or structural warnings. Meteorological Agencies report the status of the hurricane, its position, and particular warnings. Business owners describe the status and needs of their stores and personnel. Volunteers share their activities and look for critical needs. The information produced and consumed in this domain is dynamic and unpredictable, and agencies have their own protocols and formats of sharing data, for example, the Miami-Dade County Emergency Office publishes hourly document reports. Further, learning the schema from previous disasters is hard, as new situations, needs, and requirements arise.

In Fig. 1a, we show a report extracted from the National Hurricane Center repository, describing the status of a hurricane event in 2008. The report gives the current storm location, wind speed, warnings, category, advisory identifier number, and the date it was disclosed. Even though this is a text document, it contains implicitly many attribute names and values, for example, (Storm Category, 3). If we had these values properly annotated (e.g., as in Fig. 1b), we could improve the quality of searching through the database. For instance, Fig. 1c shows three sample queries for which the report of Fig. 1a is a good answer and the lack of the appropriate annotations makes it hard to retrieve it and rank it properly.

The goal of CADS is to encourage and lower the cost of creating nicely annotated documents that can be immediately useful for commonly issued semi structured queries such as the ones in Fig. 1c. 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. In our scenario, the author generates a new document and uploads it to the repository. After the upload, 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 (query workload), and the most probable attribute values given the document text. The author (creator) can inspect the form, modify the generated metadata as necessary, and submit the annotated document for storage. We should note that inserting fielded metadata is not the only scenario in which the CADS strategies are applicable.

Consider the case of processing the documents after the hurricane, to identify and extract important metadata from the documents, so that this information can be used efficiently in the future (e.g., using a Dataspaces approach). If we use automated information extraction (IE) algorithms to extract targeted relations from the document (e.g., addresses of evacuated buildings), it is important to process only documents that actually contain such information: when we process documents that do not contain the targeted information and we use automated information extraction algorithms to extract such fields, we often face a significant number of false positives, which can lead to significant quality problems in the data [4]. Similarly, if the documents are processed by humans (i.e., where there is low probability of false positives), asking humans to inspect documents, where no relevant information is present, is expensive and counterproductive. For example, if only 1 percent of the documents contain information about the address of evacuated buildings, it is going to be unnecessarily expensive to ask humans to inspect all documents to identify such information: It is much better to target and process only promising documents, with high probability of containing relevant information.

II PROBLEM STATEMENT

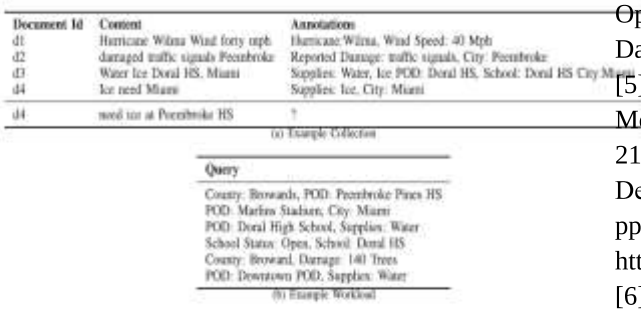
Large number of organizations today generates and share textual descriptions of their products, services, and actions. Such collections of textual data contain significant amount of structured information, which remains buried in the unstructured text. While information extraction algorithms facilitate the extraction of structured relations, they 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.

III RELATED WORK

There are several systems that favor the collaborative annotation of objects and use previous annotations or tags to annotate new objects. There have been significant amounts of work in predicting the tags for documents or other resources (WebPages, images, videos) [13], [14], [15], [6], [1]. Depending on the object and the user involvement, these approaches have different assumptions on what is expected as an input; nevertheless, the goals are similar as they expect to find missing tags that are related with the object. We argue that our approach is different as we use the workload to augment the document visibility after the tagging process. Compared with the other approaches, precision is a secondary goal as we expect that the annotator can improve the annotations on the process. On the other hand, the discovered tags assist on the tasks of retrieval instead of simply bookmarking. Dataspaces and pay-as-you-go integration. The integration model of CADS is similar to that of dataspaces [8], where a loosely integration model is proposed for heterogeneous sources. The basic difference is that dataspaces integrate existing annotations for data sources, 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 PayGo [9] and [2] are 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 [10] is a precursor of the above projects. Content management products. Microsoft Sharepoint [11] and SAP NetWeaver [12] allow users to share documents, annotate them, and perform simple keyword queries. Hard-coded attributes can be added to specialized insertion forms. CADS improve these platforms by learning the user information demand and adjusting the insertion forms accordingly. Information extraction. Information extraction is related to this effort, mainly in the context of value suggestion for the computed attributes. (See 13] for an overview of IE.) We can broadly separate the area into two main efforts: Closed IE and Open IE. Closed IE requires the user to define the schema, and then the system populates the tables with relations extracted from the text. Our work on attribute suggestion naturally complements closed IE, as we identify what attributes are likely to appear within a document. Once we have that information, we can then employ the IE system to extract the values for the attributes. Open IE [4] is closer to the needs of CADS. In particular, Open IE generates RDF-like triplets, for example, (Gustav is category 3) with no input from the user. Open IE leads to a very large number of triplets, which means that even after the successful extraction of the attribute values, we still have to deal with the problem of schema explosion that prevents the successful execution of structured queries that require knowledge of the attribute names and values that appear within a document. In principle, we could use Open IE, and then pay-as-you-go solutions for identifying equivalency relations across attribute names; however, it is much better to deal with the problem early-on, during document generation, instead of trying to fix issues that could be prevented with proper design. The CIMPLE project [5], [6] uses IE techniques to create and manage data-rich online communities, like the DBLife community. In contrast to CIMPLE, where data are extracted from existing sources and a domain expert must create a domain schema, CADS is a data sharing environment where users explicitly insert the data and the schema automatically evolves with time. Nevertheless, the IE and mass collaboration techniques of CIMPLE can help in creating adaptive insertion forms in CADS. Schema evolution. Note that the adaptive annotation in CADS can be viewed as semiautomatic schema evolution. Previous work on schema evolution [7] did not address the problem of what attribute to add to the schema, but how to support querying and other database operations when the schema changes. Query forms. Existing work on query forms can be leveraged in creating the CADS adaptive query forms. Jayapandian and Jagadish [2] propose an algorithm to extract a query form that represents most of the queries in the database using the "querability" of the columns, while in [9] they extend their work discussing forms customization. Nardi and Jagadish [3] use the schema information to autocomplete attribute or value names in query forms. In [6], keyword queries are used to select the most appropriate query forms. Our work can be considered a dual approach: instead of generating query forms using the database contents, we create the schema and contents of the database by considering the content of the query workload (and the contents of the documents, of course). The work in USHER [11] is also related: in USHER, the system automatically decides which questions in a survey are the most important to ask, given past experience with the completion of past surveys. In a sense, USHER is complementary to CADS: once we identify the attributes and values in the documents using CADS, we can then use USHER to model the dependencies across attributes and minimize the number of questions asked. Probabilistic models. Probabilistic tag recommendation systems [12], [13] have a similar goal like our system. However, the main difference is that we use the query workload in our model, reflecting the user interest.

ATTRIBUTES SUGGESTION

In this section, we study and propose solutions for the "attributes suggestion" problem. From the problem definition we identify two, potentially conflicting, properties for identifying and suggesting attributes for a document d: First, the attributes must have high querying value (QV) with respect to the query workload W. That is, they must appear in many queries in W, because the frequent attributes in W have a greater potential to improve the visibility of d. Second, the attributes must have high content value (CV) with respect to dt. 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 [5], with one key difference: our model assume that attributes are generated by two processes, in parallel: 1) 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 and 2) By knowing the types of queries that users typically issue to the database, following again a (unknown to us) joint probability distribution. Of course, the problem is inherently intractable, if we consider all possible dependencies across attributes, document content, and workload: it is very difficult to estimate the full joint distribution of so many variables. Following the common practice, when estimating language models, we consider each attribute Aj independently, and we compute the k attributes that maximize p. Given W and d as the forecasts from different sources of evidence, our system (CADS) is the decision manager, with a specific prior P_4 that decides how to combine the probability estimates from multiple sources. We experiment with two approaches: we combine the information from the forecasters assuming conditional independence, given Aj; in we build an approach that assumes conditional independence among the forecasters.



Document Id	Content	Annotations
d1	Hurricane Wilma Wind forty mph	Hurricane Wilma, Wind Speed: 40 Mph
d2	damaged traffic signals Pembroke	Reported Damage: traffic signals, City: Pembroke
d3	Water Ice Doral HS, Miami	Supplies: Water, Ice POD: Doral HS, School: Doral HS City Miami
d4	Ice need Miami	Supplies: Ice, City: Miami
d4	need ice at Pembroke HS	?

(a) Example Collection

Query
County: Browards, POD: Pembroke Pines HS
POD: Marlins Stadium, City: Miami
POD: Doral High School, Supplies: Water
School Status: Open, School: Doral HS
County: Broward, Damage: 140 Trees
POD: Downtown POD, Supplies: Water

(b) Example Workload

Fig :2 Running example.

When we increase the size of the training set. As expected, the proposed strategies increase their quality when we increase the training data size. The QV line is constant because it does not use the database but only the query workload.

IV CONCLUSION

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 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. Experiments show 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 percent. 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, 2011.

[2] S.R. Jeffery, M.J. Franklin, and A.Y. Halevy, “Pay-as-You-Go User Feedback for Dataspace Systems,” Proc. ACM SIGMOD Int’l Conf. Management Data, 2008.

[3] K. Saleem, S. Luis, Y. Deng, S.-C. Chen, V. Hristidis, and T. Li, “Towards a Business Continuity Information Network for Rapid Disaster Recovery,” Proc. Int’l Conf. Digital Govt. Research (dg.o ’08), 2008.

[4] A. Jain and P.G. Ipeirotis, “A Quality-Aware Optimizer for Information Extraction,” ACM Trans. Database Systems, vol. 34, article 5, 2009.

[5] J.M. Ponte and W.B. Croft, “A Language Modeling Approach to Information Retrieval,” Proc. 21st Ann. Int’l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR ’98), pp. 275-281, http://doi.acm.org/10.1145/290941.291008, 1998.

[6] R.T. Clemen and R.L. Winkler, “Unanimity and Compromise among Probability Forecasters,” Management Science, vol. 36, pp. 767-779, http://portal.acm.org/citation.cfm?id=81610.81609, July 1990.

[7] B. Sigurbjo¨rnsson and R. van Zwol, “Flickr Tag Recommendation Based on Collective Knowledge,” Proc. 17th Int’l Conf. World Wide Web (WWW ’08), pp. 327-336, http://doi.acm.org/10.1145/ 1367497.1367542, 2008.

[8] P.G. Ipeirotis, F. Provost, and J. Wang, “Quality Management on Amazon Mechanical Turk,” Proc. ACM SIGKDD Workshop Human Computation (HCOMP ’10), pp. 64-67, http://doi.acm.org/ 10.1145/1837885.1837906, 2010.

[9] R. Fagin, A. Lotem, and M. Naor, “Optimal Aggregation Algorithms for Middleware,” J. Computer Systems Sciences, vol. 66, pp. 614-656, http://portal.acm.org/citation. cfm?id= 861182.861185, June 2003.

[10] K.C.-C. Chang and S.-w. Hwang, “Minimal Probing: Supporting Expensive Predicates for Top-K Queries,” Proc. ACM SIGMOD Int’l Conf. Management Data, 2002.

[11] G. Tsoumakas and I. Vlahavas, “Random K-Labelsets: An Ensemble Method for Multilabel Classification,” Proc. 18th European Conf. Machine Learning (ECML ’07), pp. 406-417, http://dx.doi.org/10.1007/978-3-540-74958-5_38, 2007.

[12] M. Miah, G. Das, V. Hristidis, and H. Mannila, “Standing out in a Crowd: Selecting Attributes for Maximum Visibility,” Proc. Int’l Conf. Data Eng. (ICDE), 2008.

[13] P. Heymann, D. Ramage, and H. Garcia-Molina, “Social Tag Prediction,” Proc. 31st Ann. Int’l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR ’08), pp. 531-538, http://doi.acm.org/10.1145/1390334.1390425, 2008.

[14] Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W.-C. Lee, and C.L. Giles, “Real-Time Automatic Tag Recommendation,” Proc. 31st Ann. Int’l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR ’08), pp. 515-522, http://doi.acm.org/10.1145/ 1390334.1390423, 2008.

[15] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green, “Automatic Generation of Social Tags for Music Recommendation,” Proc. Advances in Neural Information Processing Systems 20, 2008.