EXACT: Attributed Entity Extraction By Annotating Texts

Ke Chen[1], Lei Feng[1], Qingkuang Chen[1], Gang Chen[1,2], Lidan Shou[1,2]
College of Computer Science and Technology [1],
State Key Laboratory of CAD&CG [2],
Zhejiang University, China
{chenk,11321039,chenqk,cg,should}@zju.edu.cn

ABSTRACT

Attributed entity is an entity defined by its structural attributes. Extracting attributed entities from textual documents is an important problem for a variety of big-data applications. We propose a system called EXACT for extracting attributed entities from textual documents by performing explorative annotation tasks, which create attributes and bind them to tag values. To support efficient annotation, we propose a novel tag recommendation technique based on a *few-shot learning* scheme which can suggest tags for new annotation tasks given very few human-annotated samples. We also propose a document recommendation scheme to provide run-time context for the user. Using a novel attribute index, the system can generate the task-relevant attributed entities on-the-fly. We demonstrate how these techniques can be integrated behind a novel user interface to enable productive and efficient extraction of attributed entities at limited cost in human annotation.

CCS CONCEPTS

• **Information systems → Information extraction**.

KEYWORDS

Entity Extraction; Text Document; Annotation; Neural Network

ACM Reference Format:
Ke Chen, Lei Feng, Qingkuang Chen, Gang Chen, Lidan Shou. 2019. EXACT: Attributed Entity Extraction By Annotating Texts. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3331184.3331391

1 INTRODUCTION

Extracting entities from texts is an important task for information retrieval. One interesting entity extraction problem is to obtain *attributed entities* from a large collection of textual documents. An attributed entity (AE) is an entity defined by its structural attributes and represented by a bag of attribute-value pairs. For example, given three documents as following:

- d1: *Apple iPad Air 2 A1566 Wi-Fi Silver - 64GB, 9.7"*,
- d2: *iPad mini 4 (Wi-Fi, 128GB) - Space Gray*,
- d3: *Apple iPad mini 4 32GB, WiFi+Cellular Only, 7.9in, Space Gray*,

we attempt to extract the following three attributed entities:

- e1: {*Brand*:Apple, *Model*: iPad Air 2, *Style*: Wi-Fi, *Color*: Silver, *Memory*: 64GB, *Screen*: 9.7"},
- e2: {*Brand*:Apple, *Model*: iPad mini 4, *Style*: Wi-Fi, *Color*: Space Gray, *Memory*: 128GB},
- e3: {*Brand*:Apple, *Model*: iPad mini 4, *Style*: Wi-Fi+Cellular, *Color*: Space Gray, *Memory*: 32GB, *Screen* 7.9"},

where *Brand, Model, Style, Color, Memory*, and *Screen* are the attributes extracted from the documents, and the texts to the right of the colons are the respective attribute values. Attributed entity is an extension to the well-known *named entity* as the latter can be considered as the former with an attribute *name*.

Attributed entity extraction is required in a wide variety of big-data applications. For example, market analysts need entities from texts provided by third-party online shops for product sales analysis. Government agencies need entities from the textual forms collected by the customs service for precise analysis of imported/exported goods. Such applications require AE extraction from textual documents that are characterized by four features: (1) **Large dataset** The dataset contains a large collection of textual documents. (2) **One entity per document** Each document is expected to contain only one AE. (3) **No pre-defined schema** The schemas of the AEs are not available due to the open nature of the texts. They have to be defined *during* entity extraction. An AE may share common attributes with others, but may also have some optional fields. (4) **Noises** The texts may contain typos, errors, and missing attributes/values.

It is not trivial to extract AEs from large collections of textual documents. Apparently, we cannot rely solely on either rule-based or model-based methods without prior knowledge about the dataset. One may propose to solve AE extraction using *crowd-based* methods. For example, human workers have been employed to perform taxonomy/category creation [3], crowd structuring [2], and entity resolution [4]. While these methods may help to resolve AEs from texts, they cannot adequately solve our problem due to the following reasons: (1) First, crowd-based entity resolution usually assume the availability of predefined specification (or schema) on the attributes and labels. This assumption is not true for our problem. (2) Second, by decomposing a problem into numerous small and simple questions which are disseminated to the workers, the crowd-based methods require prohibitive costs in labors answering these questions. (3) Third, almost all the crowd-based methods suffer the problem of result inconsistency among multiple workers. Previous studies propose complicated reconciliation or quality control schemes [1] to resolve the inconsistency. Unfortunately, such schemes incur even more human labor in question-answering.

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331391

Observing that in many applications the data consumers are often knowledgeable about the background of the data, we argue that our problem can be better solved by employing a small group of "power users" who can *build schemas* and *extract entities* at the same time. We advocate an approach that supports such synergetic activities in an explorative annotation process. The novelty of our system is summarized as following: (1) First, the system is able to extract the schemas and detailed entities from the dataset simultaneously via annotation. (2) Second, we employ a few-shot learning model for recommending tags from the dataset for annotation at low labelling costs. (3) Third, the system provides run-time context recommendation for annotation based on relevance and utility. (4) All these techniques are integrated behind a novel user interface to enable productive and efficient extraction of attributed entities at limited cost in human annotation.

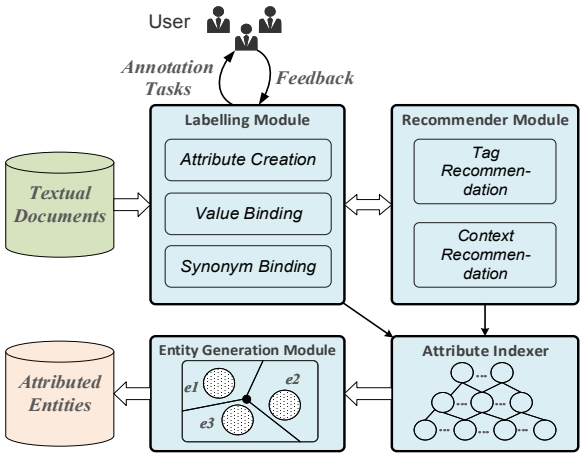

Figure 1: The EXACT System Architecture

2 SYSTEM OVERVIEW

We propose a system called EXACT (Entity eXtrACTion) to support effective AE extraction. Specifically, the system runs a server software which has access to the dataset of textual documents. A user accessing the system via a Web browser is assigned interactive *annotation tasks*, which aim at creating schemas and mapping the documents to these schemas. However, schemas are not created directly in the annotation tasks. Instead, the system allows a user to define a set of attributes for all documents, and then to produce for each document a bag of attributes which comprise the schema of this document. These attributes, in turn, must be abstracted from the values appearing in the documents. The above functionalities are realized in the iterative annotation tasks during which a user (1) recognizes new attributes from the values, and (2) binds values to the existing attributes. Enhanced by the knowledge from users, EXACT is able to create attributes, identify attribute values, and link documents to these values efficiently with the support of a few recommendation algorithms. These algorithms can effectively find representative values of an attribute with a good coverage of its value domain on the dataset, while respecting similar values containing synonyms and typos. The attributes, their values, and the respective linking information to the documents are stored and maintained in a novel index structure. Finally, the attributed entities are generated by a blocking algorithm based on the index.

Apart from the dataset and the user, EXACT contains four key components as shown in figure 1.

1) The *labelling module* which iteratively runs interactive annotation tasks. These tasks display textual objects such as tags, document snippets, and attributes in a few crafted Web pages, and collect various feedback from the user. The feedback might be to create a new attribute, to confirm a recommended tag as a value for an attribute, to select a tag in a snippet as a synonym of a value, and so forth.

2) The *attribute indexer* which dynamically captures in the index all the necessary information produced from the labelling module. The indexed information includes the attributes, the values, and the information of documents linked to each value. Each time an annotation task is submitted, the indexer takes necessary steps to update the index.

3) The *recommender module* which recommends textual objects for the annotation pages using a number of recommendation algorithms based on heuristic rules and machine-learning models.

4) The *entity generation module* which generates AEs from the indexer. As generating all entities in a single batch is expensive in computation and not user-friendly, we choose to generate entities selectively in response to each finished annotation task.

3 SYSTEM COMPONENTS

In this section, we introduce the techniques used in the four key system components.

1) The labelling module The labelling module allows for three annotation tasks, namely *attribute creation*, *value binding*, and *synonym binding*.

Attribute Creation Users can creat a new attribute by typing its name in the Web interface. Alternatively, users can select a set of tags listed in a tag box to create an attribute. It is worth mentioning that, due to the huge number of tags, EXACT has to select only a few of them for possible attribute creation and value binding. The selection of these textual objects rely on the recommender module, which will be introduced shortly.

Value Binding This task can be done in three ways. Values can be bound by manually typing tags, by selecting texts from documents directly, or by selecting from a recommended list after the user previously binds values to an attribute. The third case is usually the most productive as the recommender module can provide good recommendations based on the annotation history.

Synonym Binding This task is provided for capturing tags that are similar to (but literally differ from) the existing values of an attribute. When a value is bound to an attribute, a selection list of synonym tags are recommended to the user for possible binding of synonyms.

With properly designed UI, the labelling module allows the above tasks to be iteratively executed in any arbitrary order at the user's command.

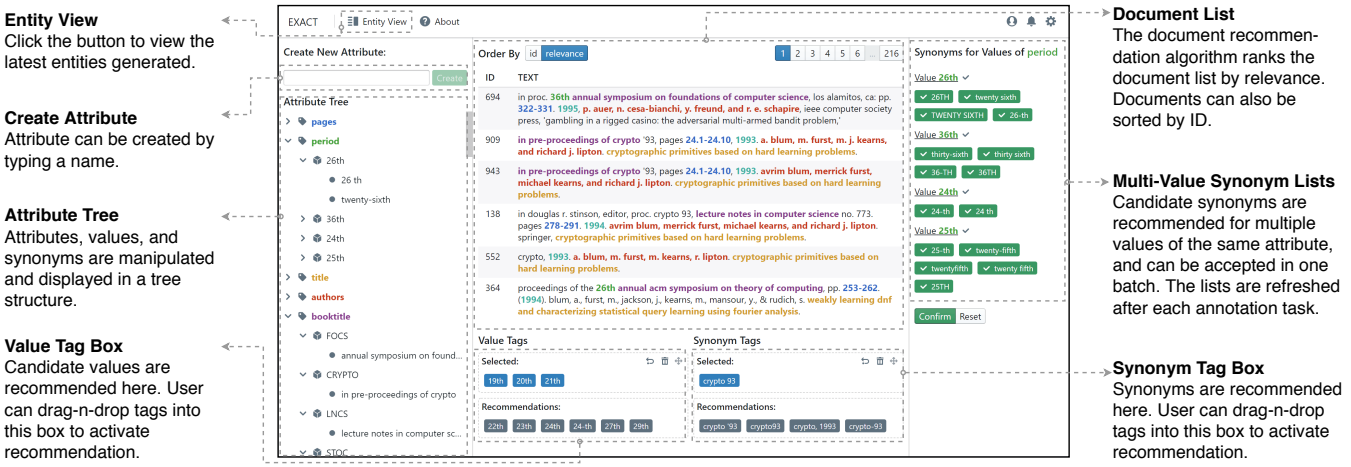

Figure 2: The Main Annotation User Interface

2) The attribute indexer The attribute indexer maintains a hierarchical index for many system operations. Generally, the index consists of a collection of layered nodes, where each layer stands for an attribute and each node $n = n(v)$ contains a set of documents $S(v)$ (actually document IDs) having the same attribute a and its respective value $a.v$. Such an index, in its raw form, might occupy large storage space. Therefore, we develop a compression technique for the attribute index to decimate its storage space. The index supports efficient operations such as insertion/deletion of a node, query by attribute/value, and set operations among multiple nodes. In the beginning of dataset processing, the index contains only one conceptual node for the entire dataset. As the interaction proceeds, new layers and nodes are added and the index grows incrementally.

3) The recommender module The design of the recommender module is key to the productivity of our system. The recommendation in EXACT is different from conventional ones as it aims at reducing the user's workload of annotation rather than learning the user's interest. We develop two categories of recommendation algorithms, namely *tag recommendation* and *document recommendation*, for the system.

Tag recommendation suggests tags as candidate values and candidate synonyms for a previously bound attribute value. As an example, when the user binds '2400CC' to attribute *SweptVolume*, tags such as '2995cc' and '2995ml' are recommended as candidate values for the same attribute. Tag recommendation can be achieved via a novel *few-shot learning* neural-network model. Briefly, given a document d and a tagged value v, the model needs to choose the most appropriate attribute $\hat{a}$; meanwhile, for the given document d and an attribute a, the model will predict an attribute value $\hat{v}$, namely a recommended tag. The model has the benefit of being able to support (attribute,value)-pair extraction without numerous training samples of (d, a, v)-tuples. We also employ a pattern-based recommendation algorithm to hide the cold-start problem with the few-shot learning.

Our document recommendation provides run-time context for the user, showing the latest annotations and recommending unannotated texts. The context must include documents that are either relevant to the latest annotations or representative of those yet to be annotated. Initially we use heuristic rules and random sampling to compute the context. Then, we employ an effcient clustering algorithm to generate the k-most representative clusters for the documents in the context, and meanwhile update the context after each round of annotation.

4) The entity generation module The entities are generated using the attribute index in a bottom-up manner. We employ a blocking algorithm [7] to split the documents of a node into subgroups. Subgroups sharing the same document IDs on different layers (attributes) indicate similar documents and may lead to a candidate entity. Each of the remaining documents of a node is evaluated a *normalized affine gap distance* [6] with the subgroups, and merged into the one with minimum distance.

For interactive viewing of entities, we cannot compute the entities once for all. Instead, we focus on the index layers that have just been updated in previous annotation tasks. We also select documents from these layers for entity generation after each annotation task.

4 IMPLEMENTATION AND EXPERIMENTS

The EXACT system was implemented using the Django Web Framework and the React Javascript library. Figure2 shows the main user interface. The *attribute tree* contains a three-tier attribute-value-synonym folder structure. A user can drag texts from the *value* or *synonym tag boxes* or even from the *document list* into the attribute tree for the annotation tasks. The two tag boxes are each populated by two areas of tags: the *selected area* and the *recommendation area*. The former includes tags selected by the user (by dragging texts from the document list or the recommended tags). The latter contains the output of tag recommendation. The novel *multi-value synonym lists* provide candidate synonyms for multiple values of an attribute. These synonyms can be accepted in one batch, as an efficient channel for synonym addition. The page also contains an "Entity View" button for viewing the new generated entities in a pop-up dialog, as shown in Figure 3. Note the AEs generated may contain duplicates, errors, and missing values, and thus need entity

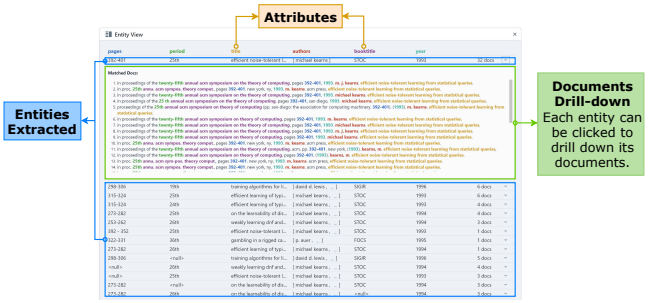

Figure 3: The Entity View Dialog

resolution/deduplication/completion. But the issues are out of the scope of this work.

For experimental study of EXACT, we use two datasets. The first is modified from Cora and contains 1295 documents, each describing a bibliographic entry for a conference paper [5]. The second is a real-world dataset of 307531 documents, each describing a bottled beer in e-commerce.

A group of three computer science students participated in the entity extraction tasks without any prior knowledge about the schemas. The first dataset was processed within 20 minutes by all users, producing 112 attributed entities in 6 attributes (*Author*, *Title*, *Booktitle*, *Period*, *Year*, *Pages*). Note that the *Author* attribute may have multiple values in each document. This does not conflict with our bag-of-attributes model. In the entity view, the multiple values are given as an array in positional order. A document is considered to be finished if at least 4 attributes are extracted. In the end of the course, documents containing less than 4 attributes are discarded. The second dataset was finished in 15 total man-hours by three users, producing 19105 entities under 14 attributes. Similarly, each document of the dataset is required to contain at least 10 attributes. Those which do not satisfy the requirement are discarded. The experimental results show that the proposed system is very promising and practical. Our preliminary performance comparison with similar systems reveals that EXACT produces competitive results while incurring much lower annotation costs.

5 THE DEMONSTRATION

We demonstrate the main use cases of EXACT in this section. These include the three annotation tasks, the tag/document recommendations involved in these tasks, and entity generation.

Use case 1: Attribute Creation We demonstrate the most productive attribute creation case. Initially the user drags texts from the document list view into the *value tag box*. The box is then populated by more tags produced from the recommender module. The user then selects the relevant values in the box and drag these to the *attribute tree*, therefore creating a new attribute. We demonstrate how the tags recommended by the few-shot learning scheme improve the productivity of the task.

Use case 2: Value Binding Upon completion of a previous attribute creation or value binding task, the system updates the recommended tags in the value tag box for binding to the same attribute.

Use case 3: Synonym Binding User can bind synonyms to a value to increase its coverage over the dataset. This is done by adding synonym tags into specific value(s) in the attribute tree. We demonstrate two scenarios: (1) First, the synonym tag box recommends tags to be bound to an existing value. (2) Second, the multi-value synonym lists provide candidate synonyms suggested for multiple values under the same attribute. For example, synonym 'twenty-sixth' and 'thirty-sixth' can be bound to value '26th' and '36th' respectively, both under attribute *Period*, in one operation.

Use case 4: Document Recommendaion As annotation tasks proceed, the document list is repeatedly updated by the document recommendation algorithm, and the documents are ranked in descending order of their relevance. Documents relevant to the latest annotation or representative of the unannotated ones have better chance to be shown in the list.

Use case 5: Entity Generation Results After a number of attributes are created and bound to values, the user can click the "Entity View" button to browse the new generated entities. Each entity can be drilled-down to view the documents from which the entity is extracted. We demonstrate that entities listed in the view are ranked by their task relevance.

6 CONCLUSIONS

We described the design of EXACT, an efficient attributed entity extraction system for large textual documents. The system allows for interactive annotation tasks which incrementally build flexible bag-of-attributes schemas and extract entities from the documents using such schemas. Equipped by the recommendation algorithms, the system is able to support efficient entity extraction at limited cost in human annotation. We believe the proposed approach can be applied in many open-world entity extraction applications. Apparently the productivity of the annotation tasks relies on the recommendations. For future work, we will focus on improving the effectiveness of the novel recommendation techniques in semi-supervised approach.

7 ACKNOWLEDGEMENTS

This work is supported by the National Basic Research Program of China (973 Program No.2015CB352400), the National Science Foundation of China (No. 61672455), the Natural Science Foundation of Zhejiang Province of China (No. LY18F020005). The work is also supported by the National Research Foundation, Prime Minister's Office, Singapore under its Strategic Capability Research Centres Funding Initiative.

REFERENCES

[1] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. Motahari-Nezhad, E. Bertino, and S. Dustdar. 2013. Quality Control in Crowdsourcing Systems: Issues and Directions. *IEEE Internet Computing* 17, 2 (2013), 76–81.
[2] Joseph Chee Chang, Aniket Kittur, and Nathan Hahn. 2016. Alloy: Clustering with Crowds and Computation. In *CHI*. 3180–3191.
[3] Lydia B. Chilton, Greg Little, Darren Edge, Daniel S. Weld, and James A. Landay. 2013. Cascade: Crowdsourcing Taxonomy Creation. In *CHI*. 1999–2008.
[4] Asif R. Khan and Hector Garcia-Molina. 2016. Attribute-based Crowd Entity Resolution. In *Proc. CIKM (CIKM '16)*. 549–558.
[5] Andrew McCallum. [n. d.]. Cora Data. https://people.cs.umass.edu/~mccallum/data.html
[6] Eugene W. Myers and Webb Miller. 1988. Optimal alignments in linear space. *Bioinformatics* 4, 1 (1988), 11–17.
[7] George Papadakis and Themis Palpanas. 2016. Blocking for large-scale Entity Resolution: Challenges, algorithms, and practical examples. In *Proc. 32nd International Conference on Data Engineering (ICDE)*. 1436–1439.