Combining Truth Discovery and Open Information Extraction with Active Ensembling

Mouhamadou Lamine BA
Qatar Computing Research Institute
Tornado Tower, West Bay
Doha, Qatar
mlba@qf.org.qa

Laure Berti-Equille
Qatar Computing Research Institute
Tornado Tower, West Bay
Doha ; Qatar
lberti.qf.org.qa

ABSTRACT

Web search engines or open information extraction systems usually reply to users' queries with a set of candidate answers that are often conflicting because they are claimed by multiple information sources. In this context, estimating information veracity is difficult for the users especially when they have no prior knowledge about the trustworthiness of the sources. In this demonstration paper, we showcase a system that supports event/entity and relation extraction based on keyword-search from the Web, processes the conflicting outputs, combines multiple truth finding algorithms with active learning to provide the most likely true answers and determine the most trustworthy sources.

1. INTRODUCTION

[**Lamine: Page allocation**]
- 1.25 pages –> abstract + introduction
- 1.5 pages –> Open information extraction + Active Ensembling for Truth Discovery
- 1 pages –> Demonstration System + Scenario
- 0.25 pages –> References

2. OPEN INFORMATION EXTRACTION

- décrire le type d'information auquel on s'intéresse par exemple "factoid claim"
- decrire le systeme sur lequel on se base
- décrire comment on transforme l'output de OpenIE
- donner qq exemples

In this study, we are interested on truth discovering over the huge number of "factoid" statements (or claims) about real-world facts returned by information extraction systems. These claims are extracted from unreliable Web sources. A "factoid" claim , e.g., *Barack Obama was born in Kenya*, is a piece of unverified or inaccurate information that is presented as factual, often as part of a publicity effort. Such type of claims is usually accepted as true because of its frequent redundancy over multiple sources. We focus on conflicting factoid claims provided by typical open Web information extraction systems as answers to users' queries. Concretly, given user input, we retrieve the set of candidates claims, together with the associated sources, returned by TextRunner [1] from a Web corpus. We then format this output in such a way that fits our truth discovering process.

Data collection. We analyze claims about real-world facts from TextRunner. TextRunner is an open Web information extraction system relying on an unsupervised extraction process on a fixed collection of Web corpus consisting of Google, ClueWeb, News, Nell, and Wikipedia corpus. Each particular corpus aggregates information from multiple other Web sources which can be of different nature, e.g. domain specific sources. When a query is issued, TextRunner first searches for the relevant corpus to query, and it then extracts the possible answers from them. To do so, it performs, for efficiency concerns, a single data-driven pass on the corpus to obtain the list of candidate relational tuples which might satisfy the user input query about a given real-world fact. In TextRunner, a user input query is typically a set of keywords consisting of two real-world *entities* and a *relation*. Such a user input query q can be defined formally as a triplet $(\mathsf{e}_1, \mathsf{r}, \mathsf{e}_2)$ where e_1 and e_2 are real-world entities and r is a relation. The argument r specifies a possible relationship that might exist between the two given entities. None of the three arguments is mandatory, meaning that some of them can be not specified by the user when issuing his query. This captures the fact the user has a partial knowledge about the real world, which is common in practice. Let denote the fact the user input query q is referred to by f_q. TextRunner will find and extract a collection of Web claims related to this fact.

TextRunner's output is thus a set of candidate claims about a specific real-world fact. This output is ranked by the system according to the number of sources that supports each claim. A claim in this context can correspond to a tuple, a relation, or an real-world entity with respect to the given keywords. TextRunner also offers the ability, through Web hyperlinks, to access to the meta-data, e.g., the source, the type, the full corpus, etc., associated to each returned claim for further exploration.

Data formatting. Given a user query q about a fact f_q, we consider the set $\mathcal{C}$ of n claims $\mathsf{c}_1, \ldots, \mathsf{c}_\mathsf{n}$ extracted by TextRunner as answers. For each claim c_i, $1 \leq i \leq n$, we go through the attached Web hyperlink and extract the set $\mathcal{S}_{\mathsf{c}_\mathsf{i}}$

1. TextRunner is accessible online at http://openie.allenai.org/

Un exemple d'extraction avec TextRunner
(a) TextRunner Extraction

Un exemple de triplés extraites
(b) Extracted Triplets

Figure 1: Data collection and formatting

of sources which support it. We achieve such a extraction by using hand-written mapping rules. We have also assumed that the system returns only one claim per source for a given query, i.e., $\mathcal{S}_{\mathsf{c}_i} \cap \mathcal{S}_{\mathsf{c}_j} = \emptyset$ for all $i \neq j$ with $1 \leq i \leq j \leq n$. We have now, for the query q about the fact $\mathsf{f_q}$, the set of claims together with the sources. In order to fit to the input of the truth finding process, we need to format the data collected from TextRunner about the facts in a certain manner. As a consequence, for the fact $\mathsf{f_q}$, we therefore consider every extracted claim $\mathsf{c_i}$ and generate a triplet $(\mathsf{f_q}, \mathsf{c_i}, \mathsf{s}_j)$ for each source $\mathsf{s}_j \in \mathcal{S}_{\mathsf{c_i}}$. We finally obtain a collection of triplets $\{(\mathsf{f_q}, \mathsf{c}, \mathsf{s}) \mid \mathsf{c} \in \mathcal{C}, \mathsf{s} \in \mathcal{S}_\mathsf{c}\}$ as the final formatting of the output of TextRunner regarding a given user input query q about a fact $\mathsf{f_q}$.

[**TODO: Quelle types de requêtes souhaitons nous supporter?**]

[**TODO: Si on considére chaque requête séparée, on se retrouve á traiter un seul claim á chaque fois. Généraliser á k requêtes utilisateurs?**]

3. ACTIVE ENSEMBLING FOR TRUTH FINDING

We describe in this section our use of active ensembling in the context of truth finding for discovering gradually the optimal set of algorithms that together maximizes the accuracy of the process when users' feedbacks arrive into the system.

3.1 Ensembling

- donner idée générale pour introduire ce qu'est l'ensembling
- on a besoin de le faire dans le contexte de truth discovery car aucune methode ne bat toutes les autres dans tous les cas de figure
- donc on combine les methodes : il y plusieurs façon de combiner par ex. consensus de méthodes, etc.
- expliquer quelles méthodes on combine avec leurs avantages et inconvénients

Ensembling, or commonly ensemble-based active learning, is a semi-supervised learning method that learns about an appropriate combinaison of multiple classifier types for a given task by interactively querying users (or other types of sources) for *labeled examples*.

In our setting, we are interesting on discovering the optimal combinaison among several possible truth finding algorithms. This combinaison is excepted to maximize the precision of the truth finding process on claims outputted by TextRunner as possible answers with respect to a user's query about a given real-world fact. A well known lack, e.g., see in [3, 5], of existing truth discovering algorithms is that they are mostly sensitive to given application domains and data characteristics. As a consequence, there is no approach that outperforms the others on all types of datasets. Furthermore, the truth finding process is harder in practical scenarios in the sense that there exists usually no labeled examples, or *ground truth data*, against which one can evaluate the precision of the different algorithms in order to choose the more accurate one on a data of interest. However, users have often some knowledge background of the truth about certain real world facts. These valuable feebacks from the user could serve as partial gold standard in order to estimate the accuracy of each truth finding algorithm. As we shall show in later, we actively query users for labeled data to use in our active approach.

An ensembling technique might enable us to combine multiple algorithm in order to obtain an optimal hybrid truth finding approach that outperform any individual approach.

Several query strategies, e.g., uncertainty sampling or query by committee, could be adopted in order to obtain from users labeled examples for the active learning process; see [4] for more details about active learning. In this study, we have used query by committee approach which is proven to be very effective in many different settings.

Our ensemble-based active learning is performed by considering twelve well established truth finding algorithms which range from naive approaches to sophisticated ones. We briefly introduce each considered class of truth disc in the following.

1. Naive techniques: The class is essentially formed by *majority voting* which considers the truth as providing the majority of sources. This class assumes that the sources are equally reliable.
2. Iterative techniques
3. EM based techniques
4. Dependency detection based techniques

3.2 Active Learning Strategy

- notre approche que l'on défend ici dans la démo est semi supervisée en impliquant de l'utiliseur de façon active en lui demandant s'il peut confirmer des faits (facts)
- si on a une ground truth partielle on la "rejoue" cas par cas

4. OUR DEMONSTRATION SYSTEM

4.1 System Architecture and User Interface

The architecture of our demonstration system, given in Figure 2, comprises the following three main components.

User I/O Interface. It represents the main entry point of our application for user interaction. The user I/O interface is composed by a text search area where a given user can enter its search keywords, in terms of a relation, The final result of the overall process will be also show to the users through this component. Finally, the user gives it feebacks via the user I/O interface through the button options or the form.

Information extraction module. This is the information extraction module which considers the input of the user and browsers several Web sources in order to returns the relevant

answers. In our system, we rely on TextRunner in order to extract information from Web corpus.

Truth Finding Engine. It corresponds to AllegatorTrack which contains twelve truth finding algorithms with different accuracy according to the types of claims and the characteristics of sources.

Learning Module. We have also a learning method that uses our knwoledge bases of users feedbacks. It enables to learn about the best truth finding algorithms, among the twelve, to use with respect to the type of entities or relations searched by the user.

Knowledge Base. The knowledge base contains the information used for the learning phase the truth finding procedure. These information include the true facts for some relations which have been learnt based on the feedbacks of the users. In addition, our knowledge base could be enriched with ground truth about some facts from reliable sources such as Wikipedia. Based on the knowledge base, our system has the ability to improve the accuracy of the truth finding process by learning about the best method to use or the best parameters, e.g., sources' accuracy scores, to consider for a better boostrapping of the process.

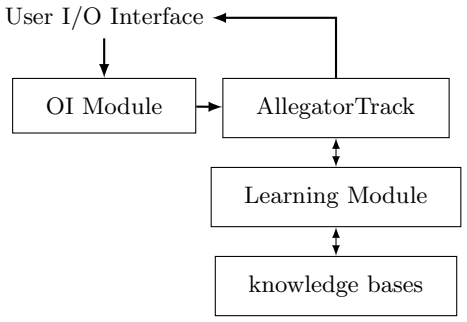

Figure 2: Architecture of our system

4.2 Demonstration Scenario

A given user that wants to interact with our system must do it through the search form. Through the search form, she (or he) provides her searched relation, e.g., "Where is born Barack Obama?". The searched relation is then passed to the information extraction engine, TextRunner system in our case, which returns a set of answers considered to be relevant for the user's request. Each claim in the returned list is processed in order to extract the corresponding sources along a detailed description of the claim which we format in a certain manner. The set of sources and the formatted versions of all claims are then passed to the truth finding module which integrate all the claims and compute the most probable answer together with the reliability scores of participated sources for the searched relation. Finally, the output of the truth finding process is returned to the user. The user can also want to review the output of our system by definitively valiiting it or not through its knwoledge of the modeled world. For example when the system has totally wrong, it may be interesting to get such a kind of feedbacks from the user in order to change the used method, as there are many available with our system, and to enhance the process for the further search about the same world. The user gives feedbacks using the option buttons on the left-hand side of the outputted claims or the text form. The feebacks given by the user is saved in knwoledge bases within our system for further processes.

5. CONCLUSION

[]

6. REFERENCES

[1] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open Information Extraction from the Web. *Commun. ACM*, 51(12):68–74, December 2008.

[2] Naeemul Hassan, Chengkai Li, and Mark Tremayne. Detecting check-worthy factual claims in presidential debates. In *Proc. CIKM*, pages 1835–1838. ACM, 2015.

[3] Xian Li, Xin Luna Dong, Kenneth Lyons, Weiyi Meng, and Divesh Srivastava. Truth finding on the deep web: Is the problem solved? *Proc. VLDB Endow.*, 6:97–108, December 2012.

[4] Burr Settles. *Active Learning*. Number 114. Morgan & Claypool Publishers,, June 2012.

[5] Dalia Attia Waguih and Laure Berti-Equille. Truth discovery algorithms: An experimental evaluation. *CoRR*, abs/1409.6428, 2014.

[6] Dalia Attia Waguih, Naman Goel, Hossam M. Hammady, and Laure Berti-Equille. AllegatorTrack: combining and reporting results of truth discovery from multi-source data. In *Proc. ICDE*, pages 1440–1443, 2015.

[7] Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. TextRunner: Open Information Extraction on the Web. In *Proc. NAACL*, pages 25–26. Association for Computational Linguistics, 2007.