J. Hlaváčová (Ed.): ITAT 2017 Proceedings, pp. 218–222
CEUR Workshop Proceedings Vol. 1885, ISSN 1613-0073, © 2017 B. Horváth, T. Horváth

Evaluating Data Sources for Crawling Events from the Web

Balázs Horváth and Tomáš Horváth

ELTE – Eötvös Loránd University, Budapest, Hungary
Faculty of Informatics, Department of Data Science and Engineering
horvath.balazs221@gmail.com, tomas.horvath@inf.elte.hu
http://t-labs.elte.hu

Abstract: The bottleneck of event recommender systems is the availability of actual, up-to-date information on events. Usually, there is no single data feed, thus information on events must be crawled from numerous sources. Ranking these sources helps the system to decide which sources to crawl and how often. In this paper, a model for event source evaluation and ranking is proposed based on well-known centrality measures from social network analysis. Experiments made on real data, crawled from Budapest event sources, shows interesting results for further research.

1 Introduction

Tourist event recommender systems need a big amount of data, preferably all events around a particular location. In order to get that data we need the organizers to upload every new event what they organize/create/host to the application which handles the data. It can be the recommender application or just a backed application where the organizers would want to be shown. If the previous solution is not acceptable or the organizers would not put enough effort to do it, then the recommender system lacks of information and cannot work as good as expected.

The other solution would be to find a feed which contains the upcoming events from each location. Unfortunately there are no feeds like that, feeds can be found about one particular topic or location's events that can be crawled as well, but do not satisfies the tourist event recommender systems need. There are almost good sources for one or two big cities in the USA, but that is not scalable if the system would expect every city or town to have their own feed like those.

The only solution for the current situation is to collect the information about the events semi-automatically from numerous sources through a data crawler engine. These event sources (denoted "sources" in the rest of the paper) can be on a different level in usefulness, some of them can be completely redundant for the system, because the same information about its events is already crawled. Others can upload informations or new events very rarely, so it is not worth to check them often. Quality differences can be discovered through the observation of the different sources. In order to save computational resources, or when a system reaches its limit, the import method have to rank sources in the queue, but how could it decide which one to rank higher? What happens if it ranks a source which played a very important role in the system very low? These sources have to be evaluated and indexed according to their importance related to our purposes.

As it is mentioned in [14], WIEN [7], XWRAP [9], STALKER [5], NoDoSe [1] and BYU [4] is a selection of the well-known often-quoted solutions for Web Data Extraction (WDE). In the past few years new approaches were published like FiVaTech [6], FiVaTech2 [2], NEXIR [12], AutoRM [13] and OXPath [15]. The last one is a wrapper language which has an optimized syntax for making the description of the WDE task easier. It also supports Javascript or CSS3 transitions, most of the modern Document Object Model (DOM) modification triggers and it can recognize Drag-and-Drop features as well. Pagination is a problem from the dynamic web pages, for that, link extraction is needed. OXPath and other solutions can handle that problem already. Unfortunately to write OXPath expressions and maintain them is costly, and involves much effort, thus it is not scaling well. DIADEM [16] utilized OXPath to give wrapper generators, which is a step closer to the right solution but they do not provider deep insight into it. An other wrapper language called NEXIR has been created for covering the whole WDE process, with pagination, data extraction and integration. The problem of scaling is not solved with wrapper languages either. FiVaTech and its improved version FiVaTech2 provide a page-level extraction approach which utilizes different DOM-based information to build up a wrapper. FiVaTech therefore utilizes tree matching, tree alignment and mining techniques to identify a template from a set of pages. FiVaTech2 improves the node recognition by including node specific features, such as visual information, DOM tree information, HTML tag contents, id-s and classes. It is clearly visible, that a ranking system is needed to be able to differentiate between solutions, ARIEX [10] is a defined framework for ranking data and information extractors and solves a specific problem, with comparing different approaches. Other missing approach is to make ranking between data sources, not the approaches. When we talk about scalability, until we do not have a general solution for the problem, we can limit the scaling by finding the way of ranking the sources and leave out the unnecessary ones. There is no such publication or solution available for the public, so we take this approach in this research. For reaching the results, a bipartite graph can be used and social network analysis methods

on it. The importance of centrality measures and social network analysis methods are discussed in [11].

An approach to event data source evaluation and ranking, using network centrality measures, is presented in the following section, followed by the description of a small proof-of-concept preliminary experiment.

2 The Proposed Model

For evaluation of sources we are considering the following attributes:

- Uniqueness of events contained in the source
- Number of events the source contains
- The importance of the source w.r.t. the other sources
- Freshness of events in the source
- Location of events contained in the source

The decision was to represent sources and events in a bipartite graph, where events and sources are both vertices and their connection is represented with edges. Thus, well-known centrality measures from social network analysis [8] can be utilized to compute the above mentioned attributes of sources.

2.1 Uniqueness

To get an indicator like uniqueness, different approaches could be considered. The first point is to find those sources, which has at least one unique event. If a source has a unique event, it is important information for the model, because it means, that if we lose that source, than we cannot get those unique events from other sources. For the purpose of finding those sources, the algorithm should go through and check the **cardinality** of each event and source as well.

For the unique event calculation, the cardinality of sources are less important than the cardinality of events. If an event can be found just in one source, that means that source is irreplaceable. Of course we cannot ignore the fact that probably the system should be able to make difference between sources which do not have unique events: It is because if one of the sources which has a lot of events, both unique and not, becomes unreachable or stops working than it is predictable that it will cause uniqueness changes in the graph.

The way of computing the uniqueness, illustrated in the algorithm 1, works as follows:

It creates a copy of the whole graph and checks for the lowest cardinality event (if there are more, then it picks a random one). It chooses one of its sources and increasing that source's uniqueness index. Then, it is going though all of the events of that source and deleting them one by one. When this step finished the source with no cardinality becomes deleted as well. These steps from picking the lowest cardinality event repeating until all the event vertices disappear from the copied graph. Then the whole loop is repeated 100 times to make the result smoother and the indicators to converge to the correct value (this step is necessary because of the random pick). In the end, to get the indicators between 0 and 1, we have to divide them with one hundred. The repetition time can be increased or decreased to make the result even smoother or make the algorithm run faster.

With this approach there will be differences between the sources which does not have any unique event, so the issue is solved with this solution.

An other issue is that sometimes to download often all the unique event holder sources the resources are not enough, that is why we need to distinguish between sources which has unique events to be able to choose the most valuable of them.

The other reason why is it needed to make a difference between unique event holder sources is, that if the sources would know the algorithm they could just try to avoid to be left out or get low ranking and they would trick the system with fake unique events. This happened with Google indexing, called black hat search engine optimization, where fake back links and meta keywords were embedded in sites to increase their position in the search results.

An approach for handling these issues is to make an additional variable, the **distinguisher**, added to the previously calculated indicators, defined as

$$distinguisher(s) = \frac{u}{uall} * \frac{1}{1+e^{-(u-\overline{u})}} \qquad (1)$$

where s denotes the source, $uall$ is the sum of all unique events in the graph, u is the sum of all unique events of the source and $\overline{u}$ is the average unique events for sources in the whole graph.

The sigmoid function in the second part of the equation handles outliers such that this step just have to distinguish between unique event holders, while not making big differences, just make a ranking. Using a sigmoid function, the differences between unique event holders are smoothed out while keeping the ranking.

2.2 Degree and Betweenness

To compute the number of events contained in the source a simple centrality measure, the degree, of the source (as a vertex in the graph) is used. Basically, the degree of the source is the number of events it contains.

An other, important, property of the source is its betweenness. It is a measure, which shows how important is the position of that particular vertex (source) in the whole network, and is computed as

$$betweenness(v) = \sum_{u \neq v \neq t} \frac{nsp_v(u,t)}{nsp(u,t)} \qquad (2)$$

where u and t are vertices not equal with v and $nsp(u,t)$ is the number of the shortest paths form u to t and the

Algorithm 1 Uniqueness

```
procedure UNIQUENESS(copy of graph)
    while size of events > 0 do
        e ← minCardinalityEvent(events)
        s ← randomPick(sources containing e)
        increaseIndicator(originalDatasource(s))
        for all a ∈ getevents(s) do
            delete(a)
        end for
        delete(s)
    end while
    indicators = {indicator(s_1), indicator(s_2), ...
                  ..., indicator(s_n)}
    for all i ∈ indicators do
        i = i + distinguisher(s)
    end for
    return indicators
end procedure
```

$nsp_v(u,t)$ is the number of shortest paths between the nodes, which goes through v vertex [17]. In our case it is used for showing how important is a source for events and find events which has high betweenness. That means that an event is connecting sources and we can observe if that is the only event which makes the source less unique or there are more of these high betweenness events in its list of events. If there is more than one of those, then we should observe if the events are connected between only the same sources, or they are distributed: it means that the source can be the connection between more sources and it can be a feed as well, which provides important information even if it does not have any unique events.

It is important to know for us which are the nodes within betweenness centrality. It is because it shows those nodes which can be concert halls, clubs or concert venues, forums, etc., collecting events of different artists. If a source is such an event collector, it can leads us to the decision, that even if it does not have unique events, it is very important for the model, because it can post new events from a new artist whose website is not crawled yet by us.

2.3 Location

From the previous properties, we can already make good measurements and propose an ranking, but there are other relevant informations, which can be important in some cases, such as keeping the data up to date or focusing on different areas or performance optimization. Location is not focusing on exact locations in this measure, just trying to decide what distance is worth to travel for the tourists.

In the preliminary experiment, using the Budapest events dataset, big part of the events are inside the smaller ring road (tram line 4–6) as illustrated in the Figure 1. For this measure we need to observe if the source is having events on the same location most of the time, or its events are at different locations, usually. If events are at the same location then the task is easy, i.e. find the relevance borders for the recommender and divide the area into circles and give points according to that. The other case is when most of the events have different locations, then the algorithm should calculate the center of the locations (carefully with the outliers) and give the score according to that.

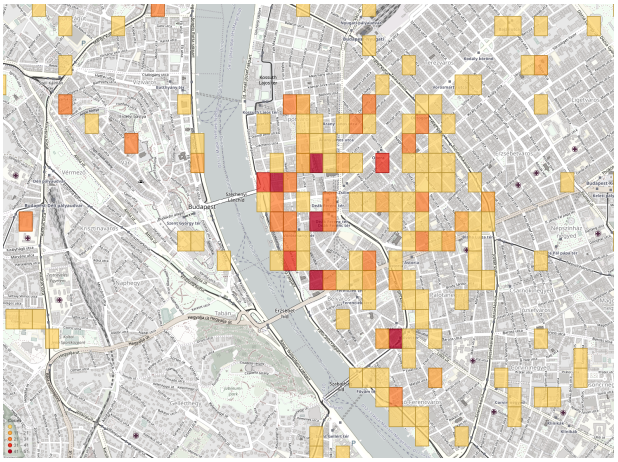

Figure 1: Locational data aggregated into hash areas in Budapest

2.4 Actuality

Freshness is a binary function [3] that measures whether the downloaded local copy is accurate according to the live page. The freshness of a page p in the repository at time t is defined as $F_p(t) = 1$ if p is equal to the local copy at time t, and, $F_p(t) = 0$, otherwise.

Age is a measure, which indicates how outdated the downloaded copy is. The age of a page p in the repository, at time t is defined as $A_p(t) = 0$ if p is not modified at time t, and, $A_p(t) = t - mt(p)$, otherwise, where $mt(p)$ is the last modification time of p.

With the help of these functions, the scheduler can calculate how often a page is usually updating the content, or in other words, how ofter is the downloaded copy gets outdated. The frequency information can tell us from different sources for the same event, which one of them posted it earlier or which one is posting more frequently. That information can influence the importance result. As an example it can be important to know if an event is canceled or changed its information like the location or the starting time. For applications where to be up to date with event informations is crucial the freshness property can be weighted more.

2.5 The Evaluation Model

This different attributes introduced in the previous chapters are aggregated to a final evaluation or ranking model of sources as follows:

$$Rank(s) = w_1 U(s) + w_2 D(s) + w_3 \frac{1}{B(s)} + w_4 A(s) + w_5 L(s) \quad (3)$$

where $w = \{w_1, w_2, w_3, w_4, w_5\}$ are the weights which will change according to the application's needs, and s is the current source what the algorithm is evaluating/ranking while U, D, B, A and L refer to the uniqueness, degree, betweenness, actuality and location of the source, respectively. The weighting is important, because there can be application which has a goal of getting all the events or as much as possible. Others can focus on performance to be able to offer trust worth fast running applications on the crawled data, and that is not harming it, if it cost some percent of the events.

3 Preliminary Experiments

251 event sources were crawled from the Web and Facebook event pages (using the Facebook Graph API), resulting in more than 1500 events (after the unification of the duplicate events). All the events crawled were from Budapest including concerts, museums, galleries, etc. The events were located mainly in the city center as can be seen in the figure 1.

The final experiments on the uniqueness part of the model were made on a dataset, where data were crawled from Facebook pages' events and clubs and museums websites. We had to consider all the possible future cases, so we made test sources as well like a complete copy of a website data, or partial copies, copies which are more important than some Facebook pages and vice versa, etc. The distinguisher is not rounded because it still should be able to make difference between sources even if the difference is smaller. In opposite of the other case where we calculate the uniqueness function on the sources, it is better to round that number, because we do not have to make too much loops to make it smoother.

A part of the result on uniqueness is illustrated on the figure 2. As can be seen, the "bjc.hu" and its copy "copyofbjc" do not have a distinguisher number, because they do not hold any unique event, obviously because they are copies of each other. So the highest number of non unique event holders is 0.5. If their event can be found in more than one other sources, then the number decreases.

Figure 3 shows an example, from of our experiments, with seven sources and their events. It is obvious that events' distances are very different from their sources. Despite these distances are not connected to the similarity of the source and the event, they represent how similar the events are. As we see in the middle, couple of the events of the big source in the middle are very far away from the others, but they are also connected to the other two sources. Those events are Jazz lessons with a famous artist and all the other events are Jazz concerts with different artists.

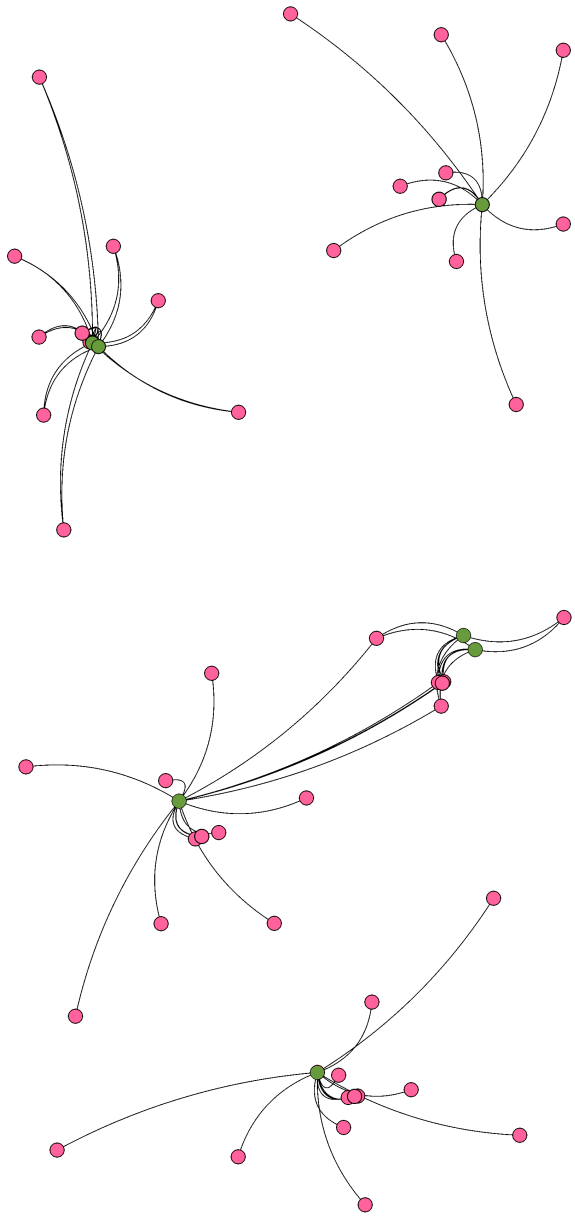

Figure 3: Visualization on 7 sources

4 Conclusions

A work-in-progress research was introduced in the paper focusing on ranking and evaluation of event data sources. The approach utilized well-known centrality measures from social network analysis what is, according to the best knowledge of the authors, the first attempt for event source evaluation.

The proposed model is quite general and can be easily modified to specific use-cases and domains. Experiments on real-world data crawled from Budapest event websites as well as Facebook pages show interesting results and promising future research directions.

```
DataSource{name='Facebook/Booty Call Thursdays'} = 1.0 + 1.9508549332420481E-5
DataSource{name='Facebook/Újszínház Budapest'} = 1.0 + 0.010709504630320549
DataSource{name='Facebook/Zrínyi Miklós Gimnázium, Budapest X.'} = 1.0 + 0.010040160260964218
DataSource{name='copyofbjc'} = 0.5 + 0.0
DataSource{name='bjc.hu'} = 0.5 + 0.0
DataSource{name='budapestbylocals.com'} = 1.0 + 0.025435073627844713
DataSource{name='eventbrite.com'} = 1.0 + 0.048862115127175365
```

Figure 2: Partial result of running the uniqueness method

Acknowledgement

Authors would like to thank T-Labs for the support and environment provided for this research. The research was conducted within the industrial project "Telekom Open City Services" supported by Magyar Telekom Nyrt.

References

[1] B. Adelberg. Nodosea tool for semi-automatically extracting structured and semistructured data from text documents. *ACM Sigmod Record vol. 27, no. 2.*, pages 283–294, 1998.

[2] C.-H. Chang C.-H. Chang and M. Kayed. Fivatech: Page-level web data extraction from template pages. *Knowledge and Data Engineering, IEEE Transactions on, vol. 22, no. 2*, pages 249–263, 2010.

[3] Carlos Castillo. *Effective Web Crawling*. PhD thesis, University of Chile, 11 2004.

[4] Y. S. Jiang S. W. Liddle D. W. Lonsdale Y.-K. Ng D. W. Embley, D. M. Campbell and R. D. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Data & Knowledge Engineering, vol. 31, no. 3*, pages 227–251, 1999.

[5] Steve Minton Ion Muslea and Craig Knoblock. Stalker: Learning extraction rules for semistructured, web-based information sources. *Proceedings of AAAI-98 Workshop on AI and Information Integration*, pages 74–81, 1998.

[6] M. Kayed and C.-H. Chang. Fivatech2: A supervised approach to role differentiation for web data extraction from template pages. *Proceedings of the 26th annual conference of the Japanese Society for Artifical Intelligence, Special Session on Web Intelligence & Data Mining, vol. 26*, pages 1–9, 2012.

[7] N. Kushmerick. *Wrapper induction for information extraction*. PhD thesis, University of Washington, 1997.

[8] Frederic Lee and Bruce Cronin. *Handbook of Research Methods and Applications in Heterodox Economics*. Edward Elgar Publishing, 2016.

[9] L. Liu, C. Pu, and W. Han. Xwrap: an xml-enabled wrapper construction system for web information sources. In *Proceedings of 16th International Conference on Data Engineering (Cat. No.00CB37073)*. IEEE Comput. Soc.

[10] R. Corchuelo P. Jimenez and H. A. Sleiman. Ariex: Automated ranking of information extractors. *Knowledge-Based Systems, vol. 93*, pages 84–108, 2016.

[11] Sebastiano Vigna Paolo Boldi. Axioms for Centrality, 2013.

[12] Y. Liu H. Wang L. Luo C. Yuan S. Shi, W. Wei and Y. Huang. Nexir: A novel web extraction rule language toward a three-stage web data extraction model. *Web Information Systems Engineering–WISE 2013. Springer*, pages 29–42, 2013.

[13] Y. Shen C. Yuan S. Shi, C. Liu and Y. Huang. Autorm: An effective approach for automatic web data record mining. *Knowledge-Based Systems, vol. 89*, pages 314–331, 2015.

[14] Andreas Schulz, Jorg Lassig, and Martin Gaedke. Practical web data extraction: Are we there yet? - a short survey. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. IEEE, oct 2016.

[15] G. Grasso C. Schallhart T. Furche, G. Gottlob and A. Sellers. Oxpath: A language for scalable, memory-efficient data extraction from web applications. *Proceedings of the VLDB Endowment, vol. 4, no. 11*, pages 1016–1027, 2011.

[16] G. Grasso O. Gunes X. Guo A. Kravchenko G. Orsi C. Schallhart-A. Sellers T. Furche, G. Gottlob and C. Wang. Diadem: domain-centric, intelligent, automated data extraction methodology. *Proceedings of the 21st international conference companion on World Wide Web. ACM*, pages 267–270, 2012.

[17] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison Wesley, us ed edition, May 2005.