SparkTrails: A MapReduce Implementation of HypTrails for Comparing Hypotheses About Human Trails

Martin Becker
University of Würzburg
becker@informatik.uni-wuerzburg.de

Hauke Mewes
University of Würzburg
hauke.mewes@stud-mail.uni-wuerzburg.de

Andreas Hotho
University of Würzburg
L3S Research Center
hotho@informatik.uni-wuerzburg.de

Dimitar Dimitrov
GESIS
dimitar.dimitrov@gesis.org

Florian Lemmerich
GESIS
florian.lemmerich@gesis.org

Markus Strohmaier
GESIS
University of Koblenz-Landau
markus.strohmaier@gesis.org

ABSTRACT

HypTrails is a bayesian approach for comparing different hypotheses about human trails on the web. While a standard implementation exists, it exposes performance issues when working with large-scale data. In this paper, we propose a distributed implementation of HypTrails based on Apache Spark taking advantage of several structural properties inherent to HypTrails. The performance improves substantially. Our implementation is publicly available.

CCS Concepts

•**Human-centered computing** → *Web-based interaction;*
•**Computing methodologies** → **MapReduce algorithms;**

1. INTRODUCTION

The Wold Wide Web is a place where users produce all kinds of trails: whether they buy products, watch the newest movies, or simply browse Wikipedia. For researchers and practitioners, it is of great interest to understand the underlying processes responsible for generating these trails. To this end, HypTrails [2] was introduced, a bayesian approach for comparing hypotheses about how trails emerge. While a standard implementation is available, it exposes performance issues when working with large-scale data. To address this, we take advantage of the structural properties of HypTrails and propose a fast, scalable and distributed implementation, called *SparkTrails*. We implement our method on Apache Spark and evaluate our approach on several large-scale datasets observing greatly improved performance and the ability to scale freely. The implementation is publicly available and open source[1].

[1]http://dmir.org/sparktrails

WWW '16 Montreal, Canada

DOI: -

2. HYPTRAILS

HypTrails [2] is a bayesian approach for comparing hypotheses about human trails on the web. Trails are modelled as a sequence of transitions between certain states. Such states can be restaurants which are reviewed in a specific order, resulting in review trails, or Wikipedia articles which are subsequently viewed, resulting in article trails (cf. [2]). HypTrails embeds these trails into a first order markov chain model. Then it calculates the evidence $P(D|H_k)$ of the data D given a hypothesis H_k incorporating the believe k in the respective hypothesis. A higher evidence signifies a better hypothesis and the larger k the more accurate the hypothesis must be to yield high evidence values. Overall, for a set of hypotheses, HypTrails calculates the evidence for several believe strengths k resulting in an evidence curve for each hypothesis. These curves are compared to judge the relative quality of the hypotheses.

The data D is represented as a transition count matrix $N = \{n_{i,j}\}$ where $n_{i,j}$ corresponds to the transition count from state i to state j, while the hypotheses are represented as stochastic matrices with each entry representing the transition probability from one state to the other. These stochastic matrices are then transformed into pseudocount matrices using the believe factor k, either by using trial roulette [2] or by multiplying each row entry by a concentration factor (e.g., the product of k and the number of states) resulting in a parameter matrix $A_k = \{a_{i,j}\}$ where $a_{i,j}$ corresponds to pseudocounts for a transition from state i to state j. Let Γ denote the gamma function, then the overall formula is:

$$P(D|H_k) = \prod_i \underbrace{\frac{\Gamma(\sum_j a_{i,j})}{\prod_j \Gamma(a_{i,j})} \frac{\prod_j \Gamma(n_{i,j} + a_{i,j})}{\Gamma(\sum_j (n_{i,j} + a_{i,j}))}}_{\text{evidence } P_i(D|H_k) \text{ for an individual state } i} \quad (1)$$

3. DISTRIBUTED IMPLEMENTATION

In this section, we cover the general idea of distributing the HypTrails method and discuss several optimizations.

Main idea. The overall evidence calculated by HypTrails corresponds to the product of the evidences of each individual state (cf. Eq. 1). Thus, we calculate these state evidences individually in a distributed fashion and merge the results into the overall evidence. The process is illustrated in Figure 1 which can be embedded into the MapReduce

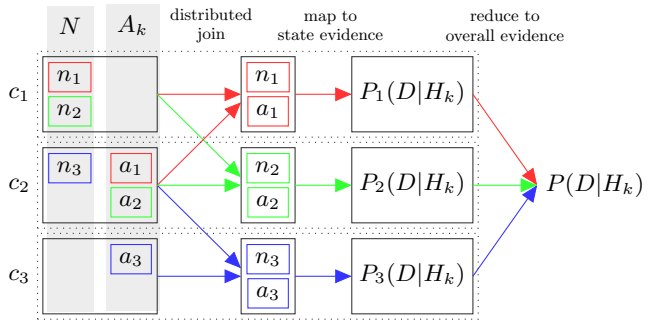

Figure 1: A schematic distributed calculation of HypTrails for three states. c_i are computational nodes where the rows of the observation matrix N and the elicited hypothesis matrix A_k are stored in a distributed fashion. After joining these two matrices by row, each computation node calculates the evidence for one (or more) state. The resulting state evidences are then merged into the overall evidence.

paradigm as indicated by the distributed join and the map and reduce steps.

Row Sparsity. Observations are often sparse resulting in many states with no outgoing transitions. For these states all $n_{i,j}$ in $P_i(D|H_k)$ are 0. Hence the components of the nominator and the denominator cancel each other out yielding an evidence of 1. Consequently these states can be left out of the distributed join (for N *and* A_k). This greatly reduces the amount of data being shuffled between nodes.

Column Sparsity. We further exploit the observation sparsity by working with sparse row vectors. For each state evidence calculation $P_i(D|H_k)$, this lets us reduce the number of Γ values to calculate by two times the number of states transitions $n_{i,j}$ which have not been observed because $\Gamma(n_{i,j} + a_{i,j})$ and $\Gamma(a_{i,j})$ cancel each other out if $n_{i,j} = 0$.

Believe. Since HypTrails calculates evidence values for several believes k, we would need to run it for each corresponding pseudocount matrix A_k separately. However, we can distribute the transition probability matrix instead of the pseudocount matrix and move the elicitation process into the state evidence calculation[2]. This results in evidence vectors, one entry for each k, avoiding expensive distr. joins.

More. Our implementation features additional optimization, such as exploiting the *row sparsity* property mentioned above for hypotheses as well, taking advantage of their structural properties to avoid data shuffling, speeding up the distributed join via pre-sorting or even consider coordinate-wise instead of row-wise calculations in case of (unlikely) memory issues. See the code base for details[1].

4. EXPERIMENTS

For evaluation we calculate the evidence for 10 different believe values k on synthetic as well as real-world data including Wikipedia navigation and phototrails [1] in Los Angeles. We test our distributed implementation based on Apache Spark and an optimized version of the original Python implementation. Table 1 lists the results for the multiplication based hypothesis elicitation variant (cf. Section 2).

SparkTrails runs on a YARN cluster with 6 worker nodes à 6 phys. Intel Xeon cores, 128GB RAM and 5 executors. The Python code is not parallelized and uses a 2.1GHz AMD Opteron CPU and 256GB RAM. For Python larger data did not fit into memory accounting for missing runtimes and we have not included the time to load data into memory (∼20 minutes for w_{nw}). For *SparkTrails* this time is included.

[2]If we choose an elicitation process which can be applied for each state independently.

Table 1: This table contains the runtimes of our experiments for real-world data from Wikipedia (w_{self}, w_{nw}) and Flickr (f) as well as several synthetic examples (s_1, s_2, r). In all cases we observe a strongly reduced runtime for the distributed algorithm (*spark*). Also, runtimes scale almost linearly when increasing the number of computation nodes (e).

	w_{self}	w_{nw}	f	s_1	s_2	r
Python	9.0m	20.1m	1.4m	-	-	-
Spark ($e = 4$)	0.4m	1.7m	3.4m	2.5h	9.7h	18.7h
Spark ($e = 8$)	0.2m	0.9m	1.7m	1.2h	4.8h	8.9h
Spark ($e = 16$)	0.1m	0.7m	1.2m	0.7h	2.7h	5.2h

For Wikipedia, observations are transitions between articles from the clickstream dataset (Feb. 2015) by Wulczyn and Taraborelli. w_{self} and w_{nw} represent the hypotheses based on the observations themselves and the link network extracted from a XML dump, respectively. While the overall state count is larger than 45 mio., the observations and the network are very sparse resulting in small runtimes. For phototrails (f), we have transitions between geo-spatial grid-cells extracted from photo sequences on Flickr; the hypothesis is based on distance. The small runtime for Python can be explained by a small state count (∼84k), sparse observations and a dense hypothesis. However, when adding the time to load data into memory (∼13m), SparkTrails is still a lot faster. To test our approach on dense data as well, we created a full transition matrix and use it as both, observations and hypothesis, with 93k (s_1) and 186k (s_2) states. Finally, we test on randomly sampled matrix with 0.01% of all entries being set for 26 mio. states (r).

Additional information on the datasets as well as the different implementations can be found online[1]. Overall, we observe that our approach, *SparkTrails*, can handle larger datasets, yields dramatically smaller runtimes, and scales well with an increased number of computational nodes.

5. CONCLUSION

We have proposed a distributed implementation of HypTrails, a bayesian method for comparing movement hypotheses on the web. Our experiments show that this implementation can handle large-scale data efficiently and outperforms non-distributed methods by a large margin. Furthermore, our approach scales almost linearly with the number of computation nodes and thus, can handle very large observation datasets and hypotheses. Future work may include efficient methods for creating large hypotheses and adapting our implementation for possible extensions to HypTrails.

6. REFERENCES

[1] M. Becker, P. Singer, F. Lemmerich, A. Hotho, D. Helic, and M. Strohmaier. Photowalking the city: Comparing hypotheses about urban photo trails on flickr. In *Social Informatics*, volume 9471 of *Lecture Notes in CS*. 2015.

[2] P. Singer, D. Helic, A. Hotho, and M. Strohmaier. Hyptrails: A bayesian approach for comparing hypotheses about human trails. In *24th Intl. World Wide Web Conf. (WWW2015)*, 2015, best paper.