TRADING ACCURACY FOR PERFORMANCE IN DATA PROCESSING APPLICATIONS

Gala Barquero[1], Javier Troya[2] and Antonio Vallecillo[1]

[1] Atenea Research Group, Universidad de Málaga, Spain

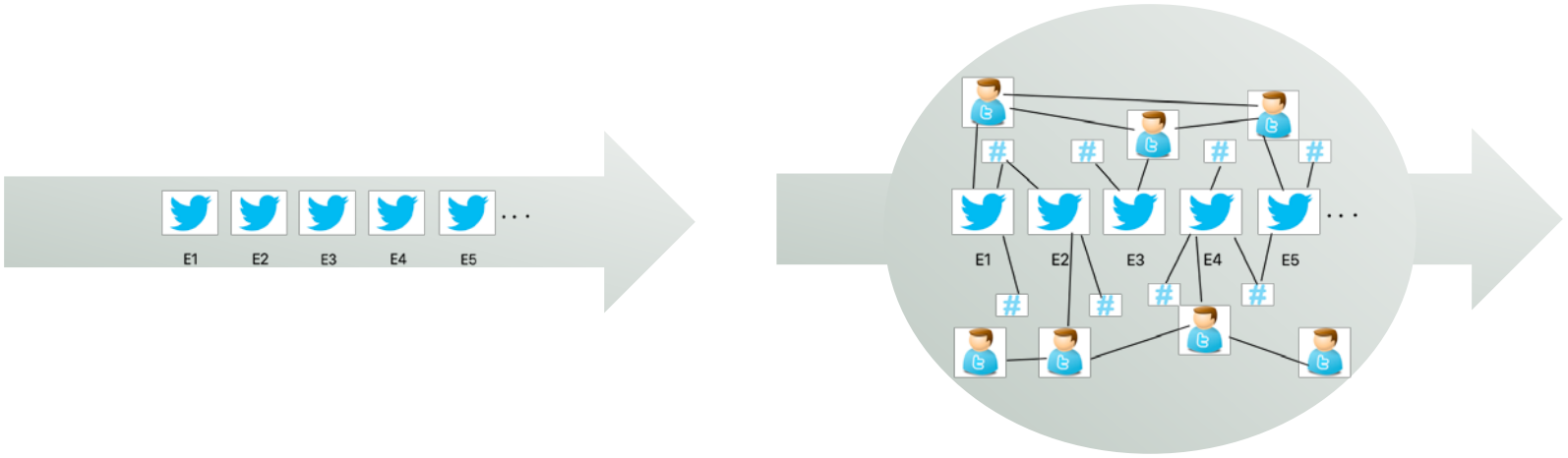
[2]ISA Group, Universidad de Sevilla, Spain

ECMFA 2019

July 15 - 19, Eindhoven, The Netherlands

Motivation

- Large amount of information flows to be processed
- Streams commonly represented as a sequence of **simple events**
- Most information sources have **interconnected events** with graph-structure:
  - Persistent information
  - Transient information



Motivation

- **Most of the data** that needs to be processed for decision making is **not significantly relevant**, particularly with large volumes of data

What is the average of the blue?

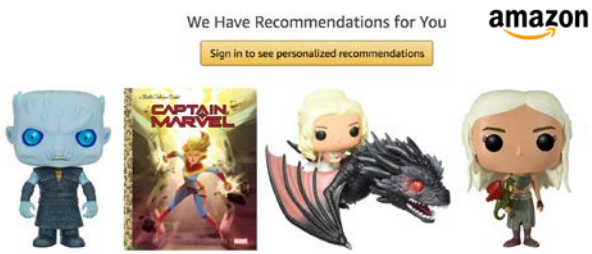
Motivation

- **Most of the data** that needs to be processed for decision making is **not significantly relevant**, particularly with large volumes of data

What is the average of the blue?

Motivation

- Most of the data that needs to be processed for decision making is not significantly relevant, particularly with large volumes of data
- Some applications **do not need extremely accurate results**, and require **fast response** times.



Purpose

- Select a subset of **data** that is **relevant** for a given query, i.e., **query/trafo approximation**

- **Estimate the error** we are making when discarding some of the input data (the one that we have considered as not relevant) in the approximation

Our Contribution

1. Analyze the **trade-off between accuracy and performance** of different types of approximations over different data distribution:
   1. Three tecniques for approximating queries over large amount of information data: **Temporal**, **Spatial** and **Random** Approximation.
   2. Consider **uniformly distributed** data vs. data **centered** on a period of time.

2. To achieve this:
   1. Apply the **Accuracy**, **Precision** and **Recall** concepts in the context of stream data processing with graph structures.
   2. Introduction of the concepts **Pattern Model**, **Approximate Model** and **Optimal Model** to work with approximations in graphs.

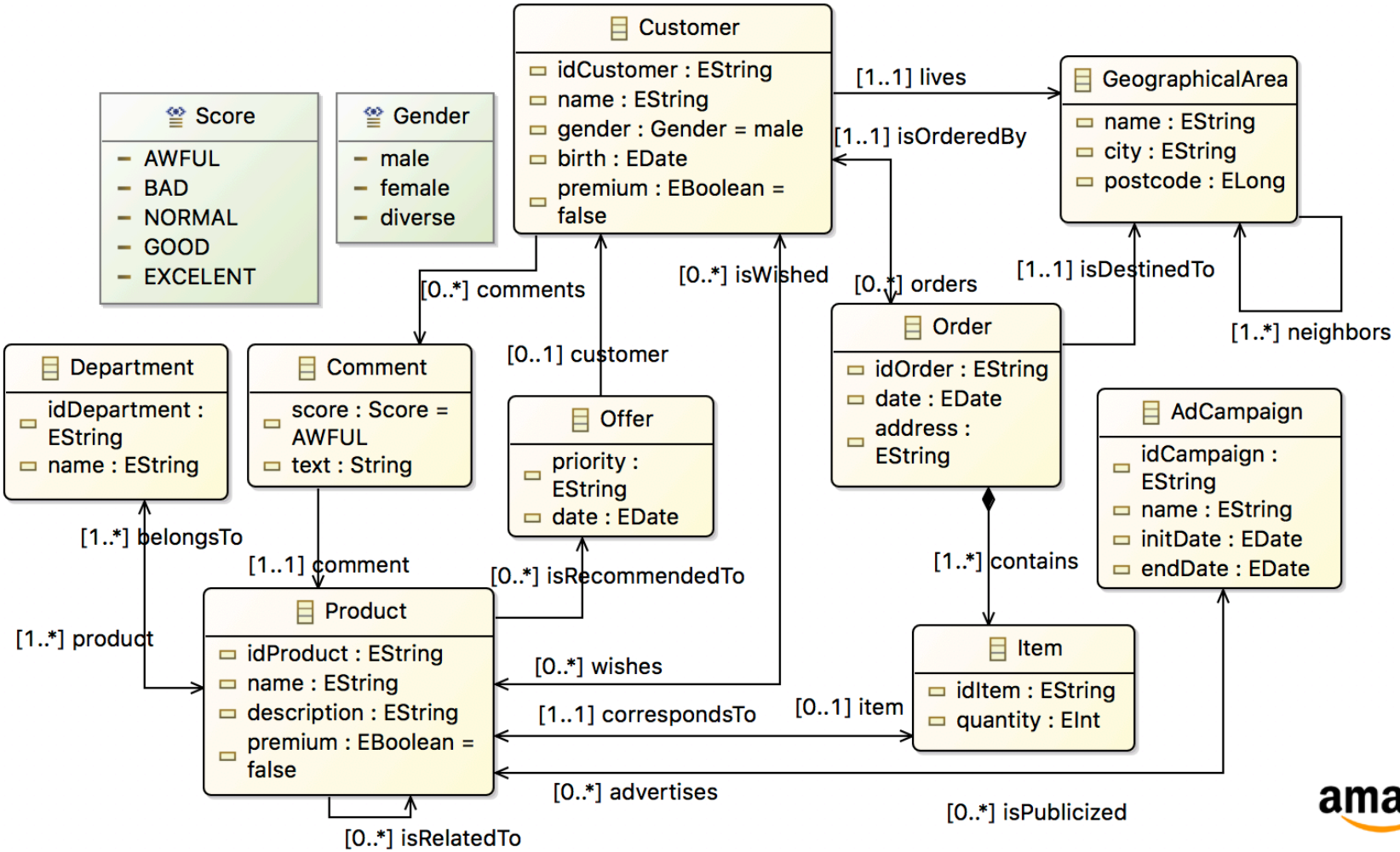
Research questions

RQ1 - How is performance improved when considering Approximate Models?

RQ2 - Can accuracy measures Precision, Recall and Accuracy help identifying the Optimal Model?

RQ3 - Which approximation method provides the best trade-off between accuracy and performance?

Case study: Amazon ordering service

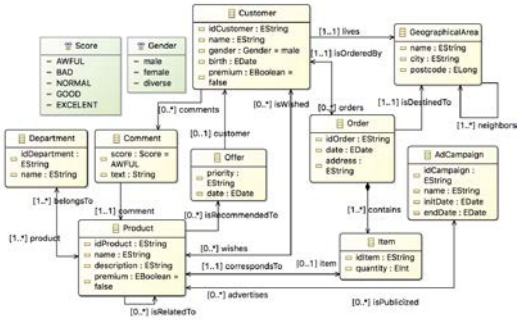


amazon

Models used for the tests

Distribution Batch	Name	Nodes	Edges
A	31K	286804	2399746
	62K	424,368	4,113,948
	125K	699,517	7,547,815
	250K	1,251,025	14,431,225
B	31K	287,731	2,477,232
	62K	425,836	4,201,686
	125K	699,945	7,635,425
	250K	1,252,316	14,543,380

Case study: Amazon ordering service

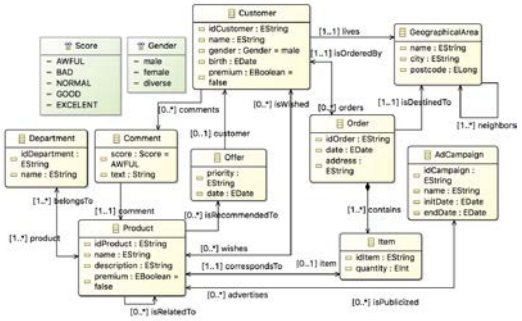


- **Q1. CreateAdCampaign:** if a product has been ordered more than 1000 times during an advertising campaign period, creates a link *isPublicized* between the product and the campaign.

- **Q2. UnpopularStock:** it returns all products that have been ordered by less than 3 customers during last month.

- **Q3. RelatedProducts:** for all the products that have been ordered last month, ckeck if there is another product included in the same order at least 100 times. The query creates a link *isRelatedTo* between both products if it does not exist.

Case study: Amazon ordering service



Q4. OlympicGamesTrending: returns the products ordered at least 100 times in Rio de Janeiro since the beginning of August 2016 until the end of the celebration of the Olympic Games. In this case, the query adds a relationship *isPublicized* between the products and the Olympic Games campaign.

Q5. RecommendsPack: if a customer has ordered *Product1* at least 5 times in different orders in the last month and this product is related to *Product2* (*isRelated* connection), then an *offer* for *Product2* is created for the customer.

NOTE: We use priorities, depending on the number of *isRelated* links that connect the two products. Here we consider only offers with priority 1 to 3.

Implementation

- TinkerGraph: in-memory graph database
- Queries with Gremlin (Related products):

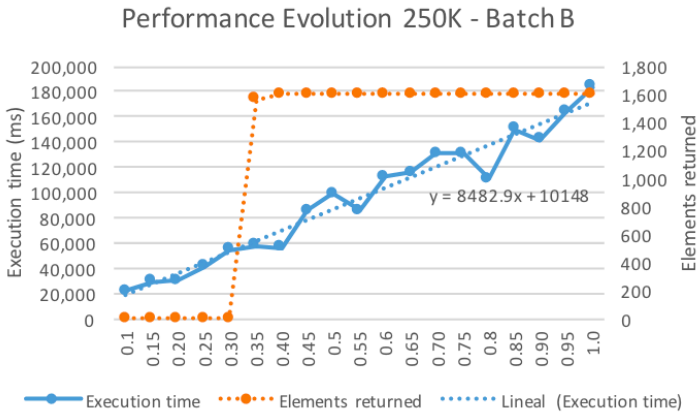
```
1  // Select Product elements
2  graph.traversal().V().hasLabel("Product").as("product1")
3  // Select Order elements that contains the products inside a temporal window
4  .in("contains").where(__.values("date").is(P.inside(initTime, endTime)))
5  // Filter orders by probability with coin step (Random approximation)
6  .coin(prob).as("order1")
7  // Select products in the same order
8  .out("contains").as("product2").where(P.neq("product1"))
9  //Check there is not a previous relationship "isRelatedTo" between products
10 .not(__.select("product1").outE("isRelatedTo").inV().where(P.eq("product2")))
11 //Count the number of matches between products and filter when they are at least 100
12 .select("product1","product2").groupCount().unfold().where(__.select(values).is(P.gte(100)))
13 // Add new elements to the graph
14 .select(keys).addE("isRelatedTo").from("product1").to("product2").iterate();
```

Implementation

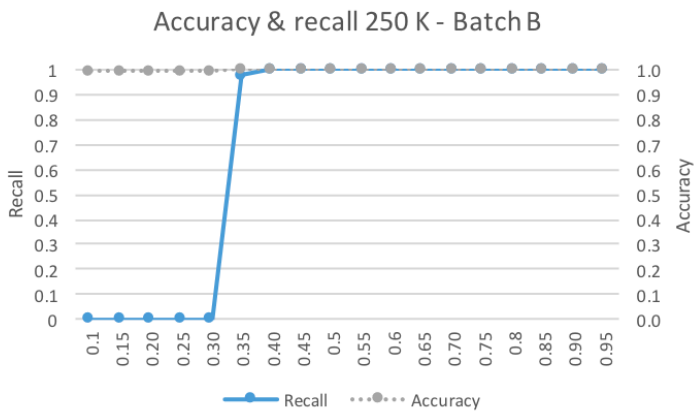
- TinkerGraph: in-memory graph database
- Queries with Gremlin (OlympicGames Campaign):

```
// Select Olympic Games campaign
graph.traversal().V().hasLabel("AdCampaign").has("name", P.eq("Olympic Games")).as("campaign")
// Take property "endDate"
.values("endDate").as("end")
// Select Geographical Area with postal code 24495L
.V().hasLabel("GeographicalArea").has("postcode", P.eq(24495L))
// Traverse the graph through relationship "neighbors" with vicinity
.repeat(__.out("neighbors")).times(hops).emit().as("area").dedup("area").select("area")
//Select orders destined to the area and ordered before "endDate" property
.in("isDestinedTo").filter(__.values("date").where(P.lte("end")))
// Select products contained by the orders
.out("contains").as("product")
//Check there is not a previous relationship "isPublicized" between products and campaign
.not(__.select("product").outE("isPublicized").inV().where(P.eq("campaign")))
//Count the number of matches between products and campaign and filter when they are at least 100
.select("campaign","product").groupCount().unfold().where(__.select(values).is(P.gte(100)))
// Add new elements to the graph
.select(keys).addE("isPublicized").from("product").to("campaign").dedup().iterate();
```

How performace improves with Approx. models

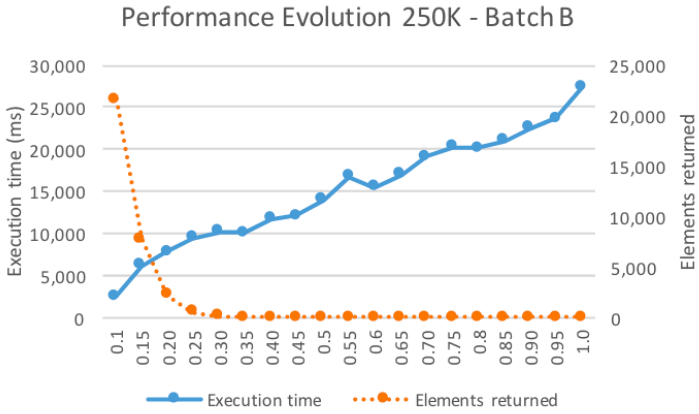


(a) Performance Evolution for Q1.

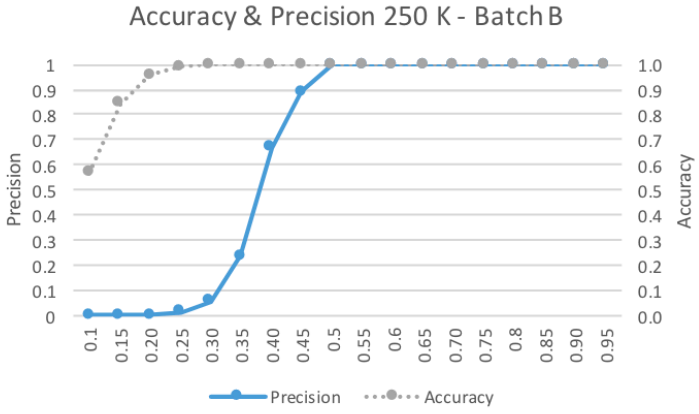


(b) Accuracy and Recall for Q1.

Example for 'CreateAdCampaign'



(c) Performance Evolution for Q2.



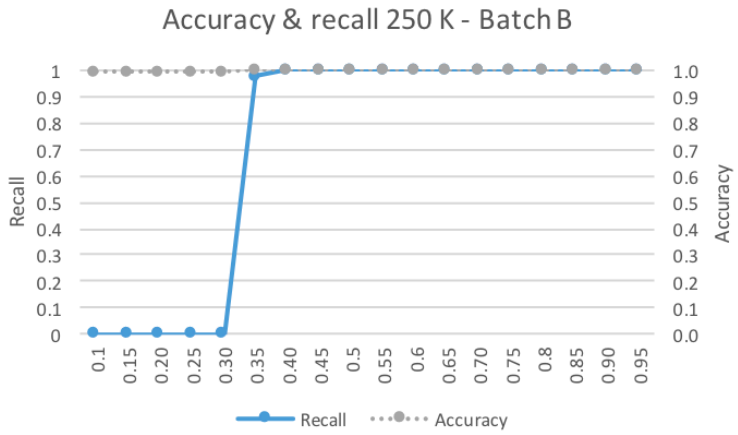
(d) Accuracy and Precision for Q2.

Example for 'UnpopularStock'

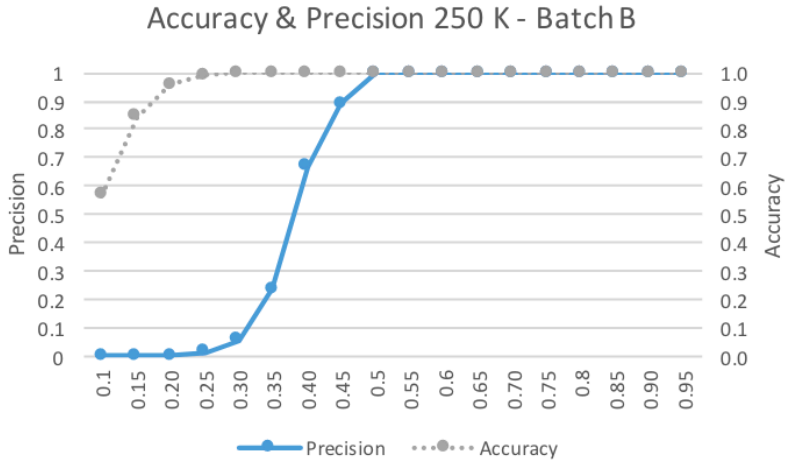
Figure 2 – Accuracy, Precision and Recall with Random Approximations.

Precision, Recall and Accuracy

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN} \qquad Accuracy = \frac{TP + TN}{TP + TN + FN + FP}$$



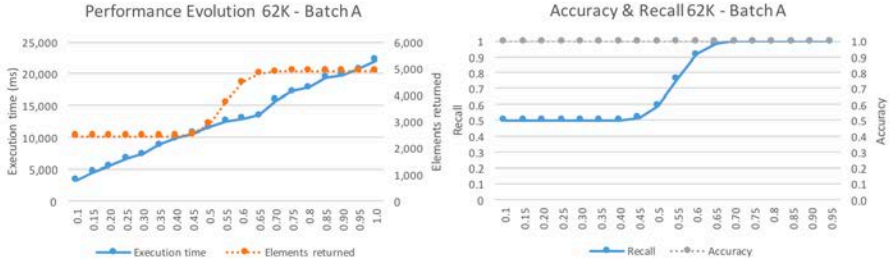
(b) Accuracy and Recall for Q1.



(d) Accuracy and Precision for Q2.

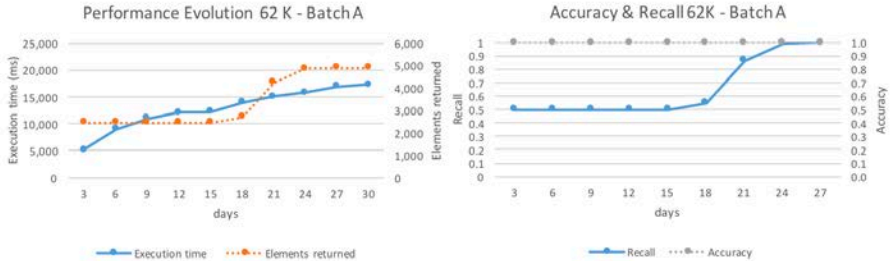
Trade-off between accuracy and performance

Data uniformly distributed



(a) Performance Evolution for Q3 with Random Approximation

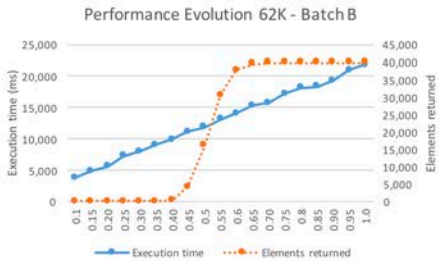
(b) Accuracy and Recall for Q3 with Random Approximation



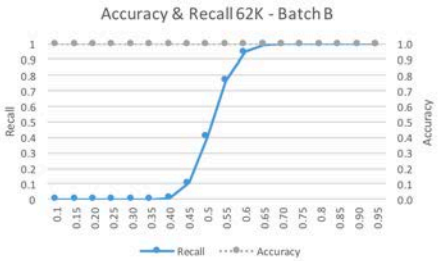
(c) Performance Evolution for Q3 with Temporal Approximation

(d) Accuracy and Recall for Q3 with Temporal Approximation

Data centered around one point in time

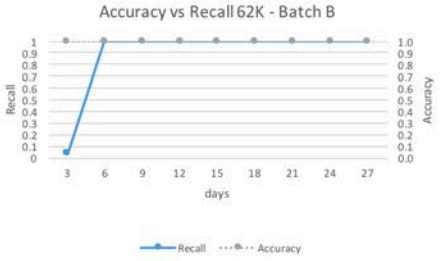


(a) Performance Evolution for Q3 with Random Approximation



(b) Accuracy and Recall for Q3 with Random Approximation

(c) Performance Evolution for Q3 with Temporal Approximation

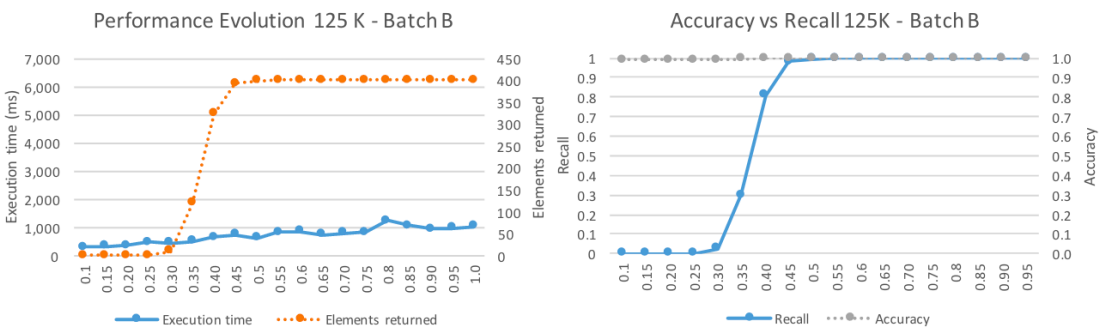


(d) Accuracy and Recall for Q3 with Temporal Approximation

Example for ‘RelatedProducts’
Random vs. Temporal

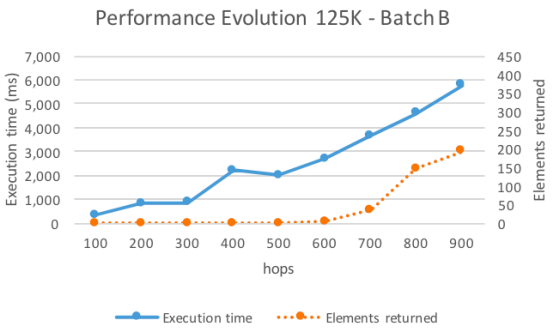
Trade-off between accuracy and performance

Data centered around one point in time

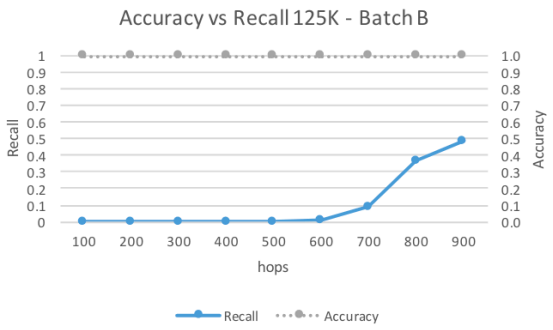


(a) Performance Evolution for Q4 with Random Approximation.

(b) Accuracy and Recall for Q4 with Random Approximation.



(c) Performance Evolution for Q4 with Spatial Approximation.

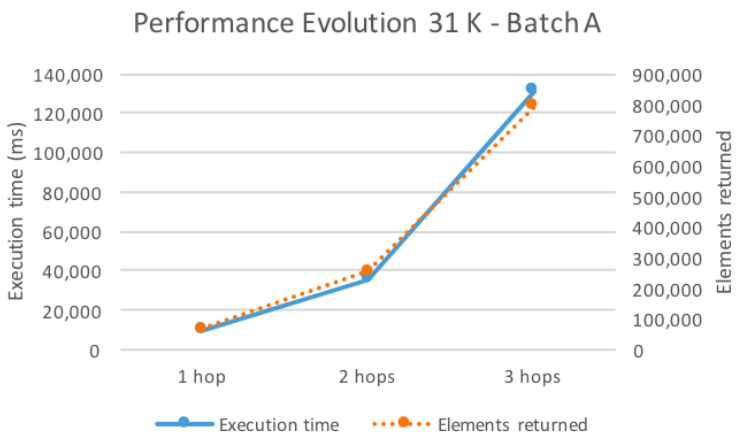


(d) Accuracy and Recall for Q4 with Spatial Approximation.

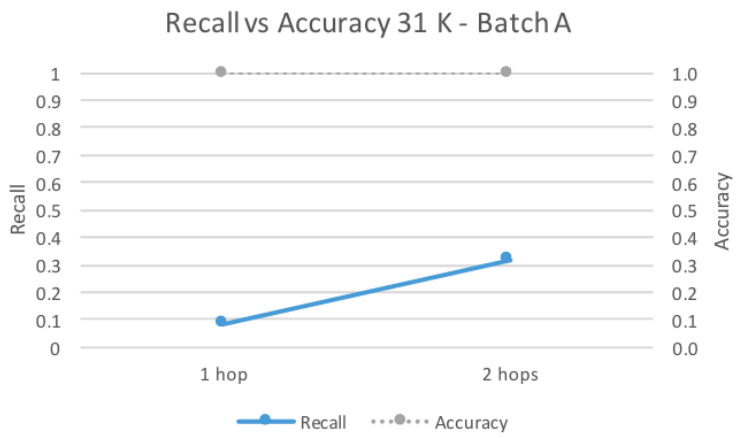
Example for ‘OlympicGame Campaign’
Random vs. Spatial

Trade-off between accuracy and performance

Data uniformly distributed



(a) Performance Evolution for Q5 with Spatial Approximation



(b) Accuracy and Recall for Q5 with Spatial Approximation

Example for ‘Recommends Pack’
Random vs. Spatial

Research questions

RQ1 - How is performance improved when considering Approximate Models?

Performance does improve (linearly) wr.t. the size of the Approximate Model vs. the size of the Pattern Model.

RQ2 - Can accuracy measures Precision, Recall and Accuracy help identifying the Optimal Model?

Accuracy is not well suited, but, precision and recall are valid measures when we get FPs and FNs, respectively.

RQ3 - Which approximation method provides the best trade-off between accuracy and performance?

No approximation method always provides the best tradeoff between performance and accuracy.

Random approximations typically behave similarly, independently from data distribution

Temporal approximations are better if data is locally centered.

Spatial approximations are expensive (time-wise) but unavoidable sometimes.

Conclusions

Contribution: analyze the tradeoff between accuracy and performance of different types of approximations over different data distribution.

- **Performance** is significantly **improved** with approximations.
- **Optimal solutions** can be found even when considering part of the source models

Data distribution	Window type in pattern model	Approximation type recommended
Uniform	No window	Random
	Temporal	Random
	Spatial	Random (spatial only if random is not possible)
Localized	No window	Random
	Temporal	Temporal
	Spatial	Random (spatial only if random is not possible)

Results of applying Random Approximations are similar no matter how the source data is distributed

Future Work

- **Data distribution:**
  - Study the presence of more than one data focus in different time intervals
  - Spatial data focus
- **Spatial approximation:**
  - Experiments with different traversal algorithms
-
- **Approximation Errors:**
  - Approximations that imply both FPs and FNs in the same query
- **Finding the Optimal Model:**
  - Method to automatically determine the subset of the source model.
  - More case studies to evaluate our proposal
  - Exhaustive evaluation about memory consumption

Thanks for your attention!

TRADING ACCURACY FOR PERFORMANCE IN DATA PROCESSING APPLICATIONS

Gala Barquero[1], Javier Troya[2] and Antonio Vallecillo[1]

[1] Atenea Research Group, Universidad de Málaga, Spain

[2]ISA Group, Universidad de Sevilla, Spain

ECMFA 2019

July 15 - 19, Eindhoven, The Netherlands

Main findings

- ✓ Random approximations are the best option when a query does not contain temporal or spatial filtering.
- ✓ Results of applying random approximations are similar no matter how the source data is distributed.
- ✓ If a query contains a temporal filter and the data is distributed with a temporal focus, then it is convenient to use a temporal approximation centered on the focus.
- ✓ If a query contains a temporal filter but the source data is uniformly distributed, then random approximations seem to perform best.
- ✓ Spatial approximations by means of hops are very expensive in terms of runtime. They are only recommended when there is no other option.