Topic-based Selectivity Estimation for Hybrid Queries over RDF Graphs

Andreas Wagner #1, Veli Bicer *2, Thanh Duc Tran #3

# *AIFB, Karlsruhe Institue of Technology*
Karlsruhe, Germany

1 `a.wagner@kit.edu`, 3 `ducthanh.tran@kit.edu`

* *IBM Research, Smarter Cities Technology Centre*
Dublin, Ireland

2 `velibice@ie.ibm.com`

***Abstract*—The Resource Description Framework (RDF) has become an accepted standard for describing entities on the Web. Many such RDF descriptions are *text-rich* – besides *structured* data, they also feature large portions of *unstructured* text. As a result, RDF data is frequently queried using predicates matching structured data, combined with string predicates for textual constraints: *hybrid queries*. Evaluating hybrid queries requires accurate means for *selectivity estimation*. Previous works on selectivity estimation, however, suffer from *inherent drawbacks*, reflected in efficiency and effective issues. In this paper, we present a general framework for hybrid selectivity estimation. Based on its requirements, we study the applicability of existing approaches. Driven by our findings, we propose a novel estimation approach, TopGuess, exploiting topic models as data synopsis. This enables us to capture correlations between structured and unstructured data in a *uniform and scalable* manner. We study TopGuess in theorical manner, and show TopGuess to guarantee a linear space complexity w.r.t. text data size, and a selectivity estimation time complexity independent from its synopsis size. In experiments on real-world data, TopGuess allowed for great improvements in estimation accuracy, without sacrificing runtime performance.**

I. Introduction

The amount of RDF on the Web is large and rapidly increasing. RDF contains descriptions of entities, with each description being a set of *triples*: $\{\langle s, p, o\rangle\}$. A triple associates an entity (subject) s with an *object* o via a *predicate* p. A set of triples forms a data graph. See Fig. 1 for a running example.

Text-rich Data. Many RDF descriptions are *text-rich*, i.e., contain large amounts of textual data. On the one hand, structured RDF often comprises text via predicates such as `comment` or `description`. Well-known examples include the DBpedia[1] or IMDB[2] dataset. On the other hand, unstructured Web documents are frequently annotated with structured data using, e.g., RDFa or Microformats.[3] Such interlinked documents can be seen as an RDF graph with document texts as objects. For instance, the Wikidata project[4] recently introduced structured data to the entire Wikipedia corpus.

Conjunctive Hybrid Queries. The standard language for querying RDF is SPARQL, which at its core features *conjunctive queries*. These queries comprise a conjunction of *query predicates* $\langle s, p, o\rangle$. Here, s, p and o may refer to a variable or a constant, i.e., an entity, a predicate or an object in the data. Consider the following SPARQL query, cf. Fig. 1 and 2:

```
SELECT * WHERE {
  ?m title "Holiday" . ?m type Movie .
  ?m starring ?p . ?p name "Audrey" .
  ?p bornIn ?l . ?l name "Belgium" .
  ?p type Person }
```

Patterns in the `WHERE` clause represent a conjunction of predicates that either match structured data (e.g., `?m starring ?p`) or *keywords* (e.g., `?p name "Audrey"`) in texts – forming a *hybrid query*. Hybrid queries are highly relevant for RDF stores with SPARQL fulltext extension, e.g., LARQ[5] or Virtuoso[6], or databases with text search, e.g., [1]. In fact, every SPARQL engine allowing `FILTER` clauses on texts has to deal with hybrid queries. For instance, `?p name "Belgium"` translates to `?p name ?name` and `FILTER contains(?name, "Belgium")`.

Selectivity Estimation. For constructing a query plan, optimizers rely on *selectivity estimates* in order to reduce intermediate results. Aiming at minimal space and time complexity, selectivity estimation techniques summarize the underlying data. Common *data synopses* include histograms [2], join synopses [3], tuple-graph synopses [4] or probabilistic relational models [5], [6], [7] (PRM). For example, a PRM approach relies on a Bayesian network (BN) as a synopsis, where predicates are modeled as random variables, each with a conditional probability distribution (CPD) associated, Fig. 3. A selectivity estimation problem is formulated by calculating the joint probability of all query predicates over a BN.

State-of-art selectivity estimation approaches exhibit a trade-off between efficiency, synopsis size and effectiveness: First, previous works restricted the dependency structure, i.e., only certain important correlations are captured in their synopsis [6], [7]. This way, synopsis size and thereby estimation time scaled to large datasets. Second, sample spaces for random variables in a synopsis are required to be small, which in turn led to a manageable overall synopsis size [5], [7]. Last, estimation via a PRM-based approach requires Bayesian inferencing to compute the query probability. It is known that such inferencing is $\mathcal{NP}$-hard. Thus, approximative inferencing techniques have been applied, trading estimation effectiveness for efficiency [5], [6], [7].

[1]http://dbpedia.org
[2]http://www.linkedmdb.org
[3]http://www.webdatacommons.org
[4]http://www.wikidata.org
[5]http://jena.sourceforge.net/ARQ/lucene-arq.html
[6]http://virtuoso.openlinksw.com

Such trade-offs are aggravated with regard to hybrid queries. Here, selectivity estimation must allow for large string sample spaces, due to string predicates that match any text value that *contains* their keywords. Thus, sample spaces (e.g., $\Omega(X_{name})$ in Fig. 1) must comprise all words and phrases (sequences of words) in the text. Synopsis size, however, grows exponentially w.r.t. sample space size, as a synopsis needs to capture dependencies between all elements in their sample spaces. Consider the CPD in Fig. 3: if a single word in sample space $\Omega(X_{name})$ is added, the CPD requires $|\Omega(X_{starring})| \cdot |\Omega(X_{title})|$ additional entries. In fact, every CPD comprising X_{name} grows similarly, as it also needs to capture the dependencies induced by the newly added word.

String synopses based on pruned suffix trees, Markov tables, histograms or n-grams, have been introduced to approximate the selectivity of single string predicates [8], [9], [10]. Targeting the rapid synopsis growth, in our previous work [7], we proposed the integration of such string synopses in a BN for summarizing string sample spaces, e.g., $\Omega(X_{name})$. However, one is again faced with a drastic trade-off between string synopsis size, and overall estimation efficiency/effectiveness. For instance, a string synopsis may use histograms for grouping similar strings in buckets: ["Audrey", "Belgium"], ["born", "British"], etc.. With less buckets being used, more *information is lost*, due to poor coverage of the string values to appear in the query – resulting in severe selectivity misestimates. In fact, with fewer buckets, correlations between single words respectively words and structured data are "washed out" and may not be captured in the overall data synopsis. On the other hand, with more fine-grained buckets, CPD size grows, thereby requiring more space and time for, e.g., marginalization. These effects translate to the entire data synopsis: the more accurate a string synopsis captures a string sample space, the more space as well as estimation time the overall data synopsis requires.

Topic-based Selectivity Estimation. Driven by these shortcomings, we introduce a general framework for selectivity estimation on hybrid queries. As a conceptually novel framework instantiation we present TopGuess. TopGuess utilizes *relational topic models* as a data synopsis, which extend traditional topic models to not only reflect text, but also structure elements [11], [12], [13]. So, in contrast to our previous work [7], we have only *one single synopsis for text and structure elements*. Further, at indexing time, we construct a data synopsis without any information loss, i.e., we index statistics for all words and structured data elements. Thus, TopGuess *does not require any trade-off* between (string) synopsis size and efficiency. At runtime, using a on-disk TRM synopsis, we construct a small, query-specific BN, which directly gives us a joint probability for the query. In fact, TopGuess does *not require any Bayesian inferencing* for estimating this joint probability, instead the probabilistic computation necessary can be formulated as a simple optimization problem.

Contributions. Contributions are as follows: (1) In this work, we present a general framework for selectivity estimation for hybrid queries over RDF and outline its main requirements. In particular, we discuss conceptual drawbacks of PRM-based solutions [5], [6], [7] in light of these requirements. (2) We introduce a novel approach, TopGuess, which utilizes a TRM-based synopsis, as a uniform summary for text and structure in RDF. (3) We provide a theoretical analysis of TopGuess, leading to space and time complexity bounds. That is, we show that TopGuess achieves *linear space complexity* w.r.t. text data size, and an estimation complexity, which is *independent of the synopsis size*. (4) We conducted extensive experiments on real-world data. Our results suggest that estimation effectiveness can be improved by up to 93% using TopGuess, while runtime performance remains comparable.

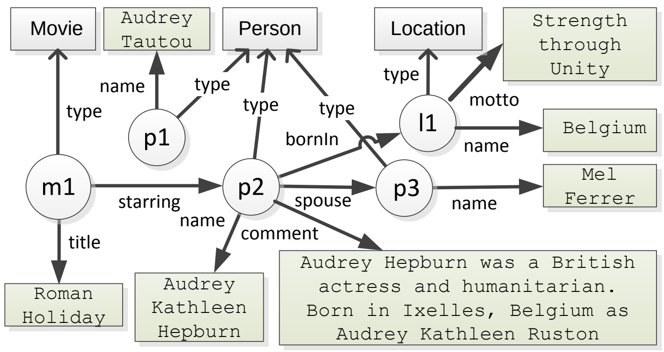


Fig. 1. RDF graph about "Audrey Hepburn" and her movie "Roman Holiday".

Outline. First, in Sect. II we outline preliminaries. Sect. III introduces a general framework for selectivity estimation. We introduce our novel TopGuess approach in Sect. IV. In Sect. V we present evaluation results, before we outline related work in Sect. VI, and conclude with Sect. VII.

II. Preliminaries

We use RDF as data and conjunctive queries as query model:

Data. Let ℓ_a (ℓ_r) denote a set of attribute (relation) labels. RDF data is given by a directed labeled graph $\mathcal{G} = (V, E, \ell_a, \ell_r)$, where V is the disjoint union $V = V_E \uplus V_A \uplus V_C$ of *entity* nodes V_E, *attribute value* nodes V_A and *class* nodes V_C. Edges (triples) $E = E_R \uplus E_A \uplus type$ are a disjoint union of *relation* edges E_R and *attribute* edges E_A. Let relation edges connect entity nodes, i.e., $\langle s, r, o\rangle \in E_R$ iff $s, o \in V_E$ and $r \in \ell_r$), and attribute edges connect an entity with an attribute value, $\langle s, a, o\rangle \in E_A$ iff $s \in V_E, o \in V_A$ and $a \in \ell_a$. The "special" edge $\langle s, type, o\rangle \in E$, $s \in V_E$ and $o \in V_C$, models entity s belonging to class o. If an attribute value $o \in V_A$ contains text, we conceive it as a *bag-of-words*. Further, we say that a vocabulary W comprises all such bags-of-words $\in V_A$. Example data is in Fig. 1.

Conjunctive Hybrid Queries. Conjunctive queries represent the basic graph pattern (BGP) feature of SPARQL. We use a particular type of conjunctive queries, *hybrid queries*: A query Q, over a data graph G, is a directed labeled graph $G_Q = (V_Q, E_Q)$ where V_Q is the disjoint union $V = V_{Q_V} \uplus V_{Q_C} \uplus V_{Q_K}$ of variable nodes (V_{Q_V}), constant nodes (V_{Q_C}) and keyword nodes (V_{Q_K}), where $o \in V_{Q_K}$ is a user-defined keyword. For simplicity, in this work we define a keyword node as "one word" occurring in an attribute value. That is, a keyword is one element from a bag-of-words representation of an attribute node. Corresponding to edge types, ℓ_a, ℓ_r, and $type$, we distinguish three kinds of query predicates: *class predicates* $\langle s, type, o\rangle, s \in V_{Q_V}, o \in V_{Q_C}$, *relation predicates* $\langle s, r, o\rangle, s \in V_{Q_V}, o \in V_{Q_C} \uplus V_{Q_V}, r \in \ell_r$ and *string predicates* $\langle s, a, o\rangle, s \in V_{Q_V}, o \in V_{Q_K}, a \in \ell_a$. Fig. 2 shows a query example. Note, a query may contain attribute predicates with other value domains, e.g., numerical values. In this work, however, we focus on string predicates.

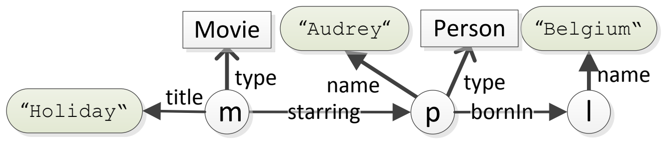


Fig. 2. Hybrid query: movies with title "Holiday" and starring "Audrey".

Query semantics follow those for BGPs. That is, results are subgraphs of the underlying data graph, which match all query patterns. The only difference is due to keyword nodes: a value node $o \in V_A$ matches a keyword $w \in V_{Q_K}$, if the bag-of-words from o contains word w. Thus, we say *string predicates have a contains semantic*.

In the following, we outline two models that we employ as selectivity estimation synopses:

Bayesian Networks. A Bayesian network (BN) is a directed graphical model allowing for a compact representation of joint distributions via its structure and parameters [14]. The structure is a directed acyclic graph, where nodes stand for random variables and edges represent dependencies. Given parents $\mathbf{Pa}(X_i) = \{X_j, \ldots, X_k\}$, a random variable X_i is dependent on $\mathbf{Pa}(X_i)$, but *conditionally independent* of all non-descendant random variables.

Example 1. *See Fig. 3-a for an BN. For instance, X_{title} and X_{movie} denote random variables. The edge $X_{movie} \rightarrow X_{title}$ refers to a dependency between the parent, $X_{movie} = \mathbf{Pa}(X_{title})$, and the child X_{title}. X_{title} is, however, conditionally independent of all non-descendant variables, e.g., X_{name}.*

BN parameters are given by conditional probability distributions (CPDs). That is, each random variable X_i is associated with a CPD capturing the conditional probability $P(X_i \,|\, \mathbf{Pa}(X_i))$. An extract of a CPD is shown in Fig. 3-b. The joint distribution $P(X_1, \ldots, X_n)$ can be estimated via the chain rule: $P(X_1, \ldots, X_n) \approx \prod_i P(X_i|\mathbf{Pa}(X_i))$ [14].

Topic Models. Topic models follow the intuition that documents are mixtures of "hidden" topics, with a topic being a probability distribution over words. These topics constitute abstract clusters of words categorized according to their co-occurrence in documents. More formally, a document collection can be represented by k topics $\mathcal{T} = \{t_i, \ldots, t_k\}$, where each $t \in \mathcal{T}$ is a multinomial distribution of words $P(w \mid t) = \beta_{tw}$ and $\sum_{w \in W} \beta_{tw} = 1$. Here, W represents the vocabulary of individual words appearing in the corpus. This way the entire corpus can be represented as k topics, which leads to a low-dimensional representation of the contained text.

Example 2. *Three topics are in Fig. 5-c. Every topic assigns a probability (represented by vector $\boldsymbol{\beta}_t$) to each word in the vocabulary, indicating the importance of the word within that topic. For instance, "Belgium" is more important in the third topic (probability $\beta_{tw} = 0.014$), than for the other two topics.*

Topic models, e.g., Latent Dirichlet allocation (LDA) [15], are modeled as graphical models, e.g., as BN. They assume a probabilistic procedure (*generative model*) by which documents can be generated. To generate a document, one chooses a distribution over topics. Then, for each word in that document, one selects a topic at random according to this distribution, and draws a word from that topic. Since the topic distribution of the documents and the word probabilities within the topics are initially unknown (hidden), this process is inverted and standard learning techniques are used to learn hidden variables and topic parameters, e.g., β_{tw}.

Problem Definition. Given a hybrid query over an RDF graph, we tackle the problem of *effective* and *efficient* selectivity estimation. We face challenges that we formally discuss in a requirements analysis, Sect. III. In the following, we outline them from an informal point of view.

- *Efficiency Issues.* A synopsis needs to capture correlations among words occurring in vocabulary W, between words and structured data, as well as among structured data elements. However, synopses, e.g., PRMs [5], [6], [7], tend to grow exponentially and become complex, in the size of vocabulary W. Consider predicate $\langle p, name,$"Audrey"$\rangle$. A synopsis may summarize the count of bindings for variable p as 2 (Fig. 1). Given a second predicate, e.g., $\langle m, starring, p\rangle$, correlations occur and a synopsis would need store the count of the conjoined query. Thus, it has to capture counts for all possible combinations with other words (e.g., "Hepburn") and structural elements (e.g., `starring`). Clearly, creating such a data synopsis is infeasible for text-rich data, as vocabulary W grows quickly with each added attribute triple. Further, with increasing synopsis size, time required by selectivity estimation techniques, such as Bayesian inferencing, commonly increases. For instance, more time is spend on marginalization over larger CPDs. We argue that this behavior is not desirable, and estimation time should be independent of vocabulary W and the synopsis.
- *Effectiveness Issues.* Space and time efficiency should not come at the expense of effectiveness. First, a vocabulary W must be accurately represented within the data synopsis. In particular, summarizing a vocabulary via a string synopsis always introduces an *information loss*. Such a loss occurs, e.g., due to eliminating "less important" words (e.g., n-gram synopsis [10]) or by capturing multiple words with one single string synopsis element (e.g., histogram synopsis [9]). Second, a synopsis should strive to capture query correlations as best as possible. That is, for a given query, an approach must reflect each query constraint. Otherwise, correlations among query predicates are not recognized and can not be exploited during estimation.

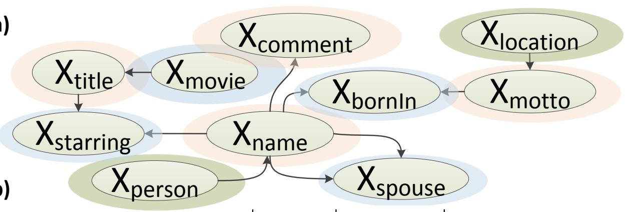
III. Selectivity Estimation Framework

In this Section, we introduce a general *framework for selectivity estimation* for hybrid queries over RDF.

Framework. Selectivity estimation strategies commonly summarize an RDF graph $\mathcal{G}$ via a concise *data synopsis*, $\mathcal{S}$, and estimate the selectivity $sel_{\mathcal{G}}(Q)$ by using *an estimation function* $F_{\mathcal{S}}(Q)$ over this synopsis. In the presence of text-rich data, a *string synopsis* function ν is defined as a mapping from the union of words comprised in attribute text values $\in V_A$ (W) and query keyword nodes $\in V_{Q_K}$ to a common representation space, denoted by $\mathcal{C}$. This common space $\mathcal{C}$ has the purpose to *compactly describe* the large set $W \uplus V_{Q_K}$, while still capturing "as much information" as possible. We define the estimation framework as:

Definition 1 (Selectivity Estimation Framework)**.** Given a data graph $\mathcal{G}$ and a query Q, an instance of the selectivity estimation framework $sel_{\mathcal{G}}(Q)$ is a tuple $(\mathcal{S}, F_{\mathcal{S}}(Q), \nu)$, where synopsis $\mathcal{S}$ represents a summary of graph $\mathcal{G}$. The estimation

(a)



(b)

$P(X_{starring} \mid X_{title}, X_{name})$	$X_{starring}$	X_{title}	X_{name}
0.111111111	T	„Roman“	„Audrey“
0	F	„Roman“	„Audrey“
0.111111111	T	„Roman“	„Hepburn“
0	F	„Roman“	„Hepburn“
0.111111111	T	„Roman“	„Audrey Kathleen“
0	F	„Roman“	„Audrey Kathleen“
0	T	„Roman“	„Tautou“
0.111111111	F	„Roman“	„Tautou“
⋮	⋮	⋮	⋮
$\sum P(X_{starring} \mid *,*) = 1$			

Fig. 3. (a) BN for example in Fig. 1. (b) Extract of CPD for $X_{starring}$.

function $F_{\mathcal{S}}(Q)$ approximates the selectivity of query Q using $\mathcal{S}$: $sel_{\mathcal{G}}(Q) \approx F_{\mathcal{S}}(Q)$. ν represents a string synopsis function defined as $\nu : W \uplus V_{Q_K} \mapsto 2^{\mathcal{C}}$.

Instantiations. The three framework components have been instantiated differently by various approaches. For instance, PRMs [5], [6], graph synopsis [4], or join samples [3] have been proposed as a data synopsis $\mathcal{S}$. Depending on the synopsis type, different estimation functions were adopted. For example, a function based on Bayesian inferencing [5], [6] or graph matching [4]. Among the many instantiations for $\mathcal{S}$ are PRMs [5], [6], [7] closest to our work.

For string synopsis function ν, approaches such as suffix trees, Markov tables, clusters or n-grams can be used [8], [9], [10]. For instance, a space $\mathcal{C}$ may comprise histogram buckets. However, most appropriate for the *contains* semantic of string predicates is recent work on n-gram synopses [10]. In particular, this approach does not limit the size of texts in the data, which is a key feature for RDF.

Combining a PRM with n-gram string synopses results in a representation space $\mathcal{C}$ comprising n-grams – resembling our previous work [7]. Here, a random variable X_a, with a as attribute, has all n-grams occurring in a's text values as sample space. Function $F_{\mathcal{S}}(Q)$ can be realized by transforming query predicates into random variables and calculating their joint probability using BN inferencing [5]. An example for such a PRM may be given as:

Example 3. *See a BN in Fig. 3-a with three kinds of random variables: class (green), relation (blue) and attribute (red) variables. Class variables, X_c, capture whether or not an entity has a particular class $\in V_C$, e.g., X_{movie} stands for class* `Movie`*. Further, a relation variable, X_r, models the event of two entities sharing a relation r. Consider, $X_{starring}$ referring to the* `starring` *relation. Each relation random variable has two parents, which correspond to the "source" respectively "target" of that relation. Class and relation random variables are binary: $\Omega(X_r) = \Omega(X_c) = \{\mathbf{T}, \mathbf{F}\}$. Last, attribute assignments are captured via random variables X_a with words as sample space. X_{title}, e.g., has words comprised in movie titles as events, Fig. 3-b.*

Requirements. Several requirements are needed for an effective and efficient framework instantiation. A synopsis $\mathcal{S}$ should summarize a data graph $\mathcal{G}$ in the "best way possible". In particular, $\mathcal{S}$ should capture correlations among words, but also correlations between words and classes/relations, **Req.1**. The synopsis size is highly important. In fact, a synopsis should aim at a linear space complexity in the size of the string synopsis, $|\mathcal{C}|$ (**Req.2**). Linear space complexity is required to eliminate the exponential growth of a data synopsis w.r.t. its string synopsis (representation space). Accordingly, the need to drastically "reduce" the vocabulary W of words $\in V_A$ via a string synopsis ν should not be necessary. This is crucial as a more compact $\mathcal{C}$ always introduces an information loss. An information loss, in turn, directly affects **Req.1** (**Req.3**). Since function $F_{\mathcal{S}}$ is used at runtime, it should be highly efficient (**Req.4**). Last, time complexity of $F_{\mathcal{S}}$ needs to be independent from size of synopsis $\mathcal{S}$ (**Req.5**).

Discussion. Closest to this paper are PRM-based approaches, e.g., [5], [6], [7]. Here, correlations are captured via probabilities in CPDs respectively a BN structure. However, when several values are assigned to the same query predicate respectively random variable representing it, an aggregation function must be applied [16]. Imagine a predicate `name` = "Hepburn" in addition to `name` = "Audrey" in Fig. 2: an aggregation would summarize both events as one single event, e.g., based on frequency of the keywords in the data. This results in one single random variable assignment. Thus, dependency information can not be reflected completely leading to misestimations, as we observed in our experiments and previous work [7] (partially fulfilled **Req.1**).

Data synopsis size of a BN is exponential in the string synopsis size, $|\mathcal{C}|$. Let a CDP be given as $P(X_i|\mathbf{Pa}(X_i))$, and assume a new word is added to sample space of $X_j \in \mathbf{Pa}(X_i)$. Then, the entire CPD grows with $|\Omega(X_i)| \cdot \prod_{X_z \in \mathbf{Pa}(X_i), z \neq j} |\Omega(X_z)|$ entries.[7] Thus, **Req.2** is not fulfilled. Further, there is the need to reduce the representation space, as the size of synopsis $\mathcal{S}$ is strongly affected by the size of $\mathcal{C}$. This, in turn, leads to an information loss (not meeting **Req.3**): in the case of an n-gram string synopsis, reducing $\mathcal{C}$ means to select a subset of n-grams occurring in the text. Note, discarded n-grams may only be estimated via heuristics [10]. The probabilities computed from those heuristics may not correspond to the actual probability of the keyword in the query predicate. Our experiments show that these information losses lead to significant estimation errors.

PRM synopses use BN inferencing for the estimation function $F_{\mathcal{S}}$ [5], [6], [7]. However, inferencing is $\mathcal{NP}$-hard [17]. Thus, "exact" computation of $F_{\mathcal{S}}$ is not feasible – instead *approximation strategies*, e.g., Markov Chain Monte Carlo methods, are used to guarantee an polynomial time complexity of $F_{\mathcal{S}}$ [14] (partially fulfills **Req.4**). Last, PRMs require expensive computation at runtime. (1) An *unrolling procedure* is needed [14], i.e., an "unrolled" BN is generated via marginalization. (2) This BN is used for inferencing to compute the query probability. For both steps, however, computation time is driven by CPD sizes, and thus, synopsis size and complexity of $F_{\mathcal{S}}$ is directly coupled, **Req.5** fails.

[7]This growth factor is referring to the worst-case. For instance, a tree-shaped CPD may reduce the additional space in some cases [14].

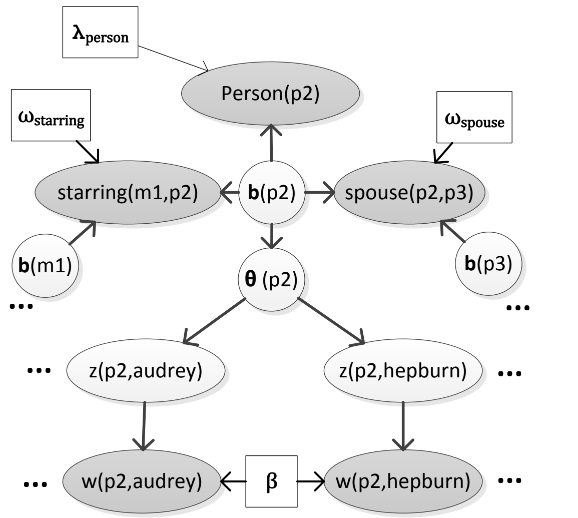


Fig. 4. TRM BN extract for entity p_2: observed variables (dark Grey), hidden variables (light Grey) and TRM parameters (rectangles). Note, relation `bornIn` is not shown for space reasons.

IV. TopGuess

TopGuess is a novel framework instantiation and adheres to several design decisions that target the above requirements. In fact, we will show that TopGuess matches all framework requirements. In the following, we present a synopsis $\mathcal{S}$ in Sect. IV-A, and an estimation function $F_{\mathcal{S}}$ in Sect. IV-B.

A. Topic-based Data Synopsis

We exploit a relational topic model [11], [12], [13] as synopsis $\mathcal{S}$, which provides us with a *single, uniform synopsis* for structured and unstructured data. More precisely, we use *Topical Relational Model* (TRM) [13], as it is tailored towards RDF data. Further, TRM parameters may be used for calculating query predicate probabilities – as we will show.

Synopsis. A TRM summarizes texts via a small set of topics, and finds correlations between those topics and structured elements. That is, a TRM assumes that if entities exhibit structural resemblances (have similar classes or relations), their words and topics respectively, shall also be similar. Vice versa, given specific words and topics respectively, some structure elements are more probable to be observed than others. For instance, one may observe that words like "Audrey" highly correlate with classes `Person` or `Movie`, as well as with other words such as "play" and "role" in the context of a relation `starring`. Thus, a TRM constitutes a *uniform synopsis* for textual as well as structural data, **Req.1**.

A TRM captures correlations between text and structure via a set of k topics $\mathcal{T} = \{t_i, \ldots, t_k\}$. Each topic $t \in \mathcal{T}$ is a multinomial distribution of words $p(w|t) = \beta_{tw}$. As before, W is a vocabulary derived from words in attribute values: for each triple $\langle s, p, o\rangle \in E_A$ we add all words in o to W. Most importantly, we do not require a string synopsis to summarize our vocabulary, i.e., we exploit the *complete vocabulary* W. Thus, W equals the common representation space $\mathcal{C}$, **Req.1+3**. This can be achieved by an on-disk storage of the synopsis. That is, as opposed to Bayesian inferencing implementations, the TRM estimation function does not require an in-memory synopsis (explained later).

In its learning process, a TRM is modeled as a BN based on information from the underlying data graph:

Example 4. *Fig. 4 depicts an extract of a TRM BN constructed for entity p_2 from our example in Fig. 1. Observed variables (dark Grey) consist of entity words (e.g., $w(p_2$,"Audrey"$)$, entity classes (e.g., $\texttt{Person}(p_2)$) and entity relations (e.g.,$\texttt{starring}(m_1, p_2)$). Dependencies among observed variables are reflected by a set of hidden variables (light Grey), which are initially unobserved, but inferred during learning: the variable $\boldsymbol{b}(p_2)$ is a topic vector indicating the presence/absence of topics for entity p_2. Note, a relation variable also depends on a variable $\boldsymbol{b}$, modeling the other entity involved in the relation. This way $\boldsymbol{b}(p_2)$ "controls" topics selected for an entity according to structure information. In addition, $\theta(p_2)$ models the topic proportion according to $\boldsymbol{b}(p_2)$, whereas every variable $z(p_2, *)$ selects a particular topic for each word by sampling over $\theta(p_2)$.*

A TRM is constructed using a generative process, which is controlled via its three parameters: $\boldsymbol{\beta}$, $\boldsymbol{\lambda}$, and $\boldsymbol{\omega}$ (shown in rectangles in Fig. 4). Hidden variables as well as parameters are inferred via a variational Bayesian learning technique as an offline process. Further discussion on the construction is unfortunately out of scope. For this work, we apply standard TRM learning as presented in [13].

Important for selectivity estimation, however, are solely the learned TRM parameters that specify the topics $\mathcal{T}$ and also qualify the degree of dependency between structured data elements and topics:

- *Class-Topic Parameter* $\boldsymbol{\lambda}$. A TRM captures correlations between classes $\in V_C$ and hidden topics $\in \mathcal{T}$ via a global parameter $\boldsymbol{\lambda}$. $\boldsymbol{\lambda}$ is represented as a $|V_C| \times k$ matrix, where each row λ_c ($c \in V_C$) is a topic vector and each vector element λ_{ct} represents the weight between class c and topic t.
- *Relation-Topic Parameter* $\boldsymbol{\omega}$. Given k topic $\in \mathcal{T}$, the probability of observing a relation r is captured in a $k \times k$ matrix ω_r. For any two entities s/o, such that s is associated with topic t_k and o with topic t_l, the weight of observing a relation r between these entities $\langle s, r, o\rangle$ is given by an entry (k, l) in matrix ω_r (denoted by $\omega_{rt_kt_l}$). Note, a TRM provides a matrix ω for each distinct relation in $\mathcal{G}$.

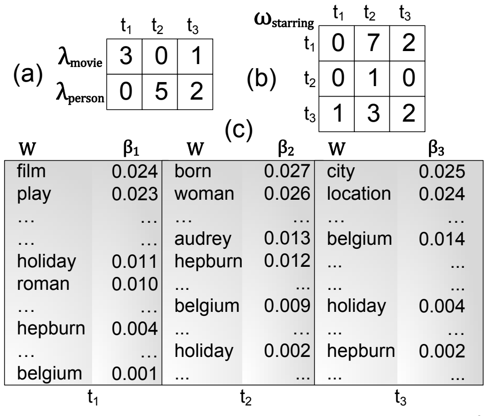
Above TRM parameters are shared among all entities in the data graph. See Fig. 5 for an example.

Now, let us formally show that TopGuess holds **Req.2**:

Theorem 1 (Synopsis Space Complexity)**.** Given k topics and a vocabulary W, a TRM requires a fixed-size space of the order of $O(|W| \cdot k + |V_C| \cdot k + |\ell_r| \cdot k^2)$.

Proof. For each topic, we store probabilities of every word in W, so the complexity of K topics is $O(|W| \cdot K)$. $\boldsymbol{\lambda}$ can be represented as a matrix $|V_C| \times K$, associating classes with topics $\in O(|V_C| \cdot K)$. Every relation is represented as a matrix $K \times K$, resulting in a total synopsis space complexity $O(|W| \cdot K + |V_C| \cdot K + |\ell_r| \cdot K^2)$ ■

Discussion. As a data synopsis $\mathcal{S}$ for selectivity estimation a TRM offers unique characteristics: First, learned topics provide a *low-dimensional data summary*. Depending on the complexity of the structure and the amount of the textual data, a small number of topics (e.g., 50) can easily capture meaningful correlations from the data graph. Notice, while we "manually" set the number of topics for our evaluation system,



(a)

	t_1	t_2	t_3
λ_{movie}	3	0	1
λ_{person}	0	5	2

(b)

$\omega_{starring}$	t_1	t_2	t_3
t_1	0	7	2
t_2	0	1	0
t_3	1	3	2

(c)

W	β_1	W	β_2	W	β_3
film	0.024	born	0.027	city	0.025
play	0.023	woman	0.026	location	0.024
…	…	…	…	…	…
…	…	audrey	0.013	belgium	0.014
holiday	0.011	hepburn	0.012	...	...
roman	0.010	...	...	...	...
…	…	belgium	0.009	holiday	0.004
hepburn	0.004	...	…	...	…
…	…	holiday	0.002	hepburn	0.002
belgium	0.001	...	...	...	...
t_1		t_2		t_3	

Fig. 5. TRM parameters for three topics: (a) Unnormalized λ_{movie} and λ_{person} parameters over three topics. (b) ω matrix for `starring` relation (rows (columns) represent source (target) topics of the relation). (c) Selected words in three topics with their corresponding probabilities. Note, data is taken from the running example, cf. Fig. 1.

our approach could be extended to learn the optimal number of topics via a non-parametric Bayesian model [18]. By means of this low-dimensional summary, a TRM provides a synopsis with linear space complexity w.r.t. the string synopsis (to be precise, linear in vocabulary W), see Theorem 1, **Req.2+3**.

Second, each topic has a *broad coverage*, as every word in the vocabulary is covered in each topic with distinct probabilities, cf. β_t in Fig. 5-c. Thus, in contrast to synopses based on graphical models, e.g., PRMs [14], a TRM is not restricted to small sample spaces. We exploit this coverage later, as a *query variable is conceived as a mixture of topics*. Thereby allowing fine-grained dependencies between words and structure in a query to be captured, **Req.1**.

Maintenance. For a TRM-based synopsis dealing with changes in the data is two-fold. First, in case of minor variations, TRM parameters may be still allow for accurate selectivity estimation. This is due to the fact that a TRM captures dependencies between text and structured elements via probability distributions over topics. We observed in our experiments that such probability distribution are invariant given small changes in the data. We learned TRMs from different samples (sizes) of the underlying data, however, the resulting selectivity estimations respectively topic distributions hardly differed. Second, in case of major changes in the data, TRM probability parameters must be recomputed. In our experiments, TRM construction could be done in under 5h. However, recent work on topic models [19] has shown that this learning process can be parallelized, thereby guaranteeing a scalable TRM construction even for large data graphs. Furthermore, [20] introduced an algorithm for incremental topic model learning over text streams. Both such directions may be exploited in future work.

B. Selectivity Estimation Function

The estimation function $F_S(Q)$ can be decomposed as [5]:

$$F_S(Q) = \mathcal{R}(Q) \cdot \mathcal{P}(Q)$$

Let $\mathcal{R}$ be a function $\mathcal{R} : Q \to \mathbb{N}$ providing an *upper bound cardinality* for a result set for query Q. Further, let $\mathcal{P}$ be a *probabilistic component* assigning a probability to Q that models whether or not its result is non-empty. $\mathcal{R}(Q)$ can be easily computed as product over "class cardinalities" of Q [5]. That is, for each variable $v \in V_{Q_V}$ we bound the number of its bindings, $R(v)$, as number of entities belonging v's class: $|\{s|\langle s, type, c\rangle \in E\}|$. If v has no class, we use the number of all entities, $|V_E|$, as bound. Then, $\mathcal{R}(Q) = \prod_v R(v)$.

For the probabilistic component $\mathcal{P}$, we first outline the construction of a query-specific BN, and afterwards show estimation of $\mathcal{P}(Q)$ by means of this BN.

Query-Specific BN. A query-specific BN follows the same intuition as an "unrolled" BN in a template-based graphical model, e.g., a PRM [14]. In both cases, one constructs a small BN at runtime for the current query by using probabilities and dependency information from a query-independent data synopsis. However, our query-specific BN differs: (1) Query dependencies are modeled very fine-grained, as each query variable is captured as a mixture of topics. (2) It follows a simple, yet effective fixed structure (topical independence assumption). In particular, our BN comprises one random variable for each query predicate. Thus, multiple assignments to a random variable can not occur – aggregations are not needed. (3) Construction does not require marginalization, instead TRM parameters can be used directly, **Req.4**.

Let us present the query-specific BN in more detail. We capture every query predicate as a random variable: for a class $\langle s, type, c\rangle$ and relation predicate $\langle s, r, o\rangle$, we introduce a binary random variable X_c and X_r, respectively. Similarly, for a string predicate $\langle s, a, w\rangle$, we introduce a binary random variable X_w.[8] Further, every query variable $v \in V_{Q_V}$ (e.g., m, p and l in Fig. 2) is considered as referring to a topic in the TRM and introduced via a *topical random variable*, X_v. However, instead of a "hard" assignment of variable X_v to a topic, X_v has a multinomial distribution over the topics. Thus, X_v captures query variable v's "relatedness" to every topic:

Definition 2. For a set of topics $\mathcal{T}$, a query Q and its variable $v \in V_{Q_V}$, the random variable X_v is a multinomial *topical random variable* for v, with topics $\mathcal{T}$ as sample space.

Based on topical random variables, we perceive query variables as topic mixtures. Then, we establish dependencies between topical random variables and random variables for class (X_c), relation (X_r) and string predicates (X_w). In order to obtain a simple network structure, we employ a fixed structure assumption:

Definition 3 (Topical Independence Asmp.)**.** Given a query Q and its variables V_{Q_V}, the probability of every query predicate random variable, X_i, $i \in \{w, c, r\}$, is independent from any other predicate random variable, given its parent topical random variable(s), $\mathbf{Pa}(X_i) \subseteq \{X_v\}_{v \in V_{Q_V}}$.

The topical independence assumption lies at the core of the TopGuess approach. It considers that query predicate probabilities depend on (and governed by) the topics of their corresponding query variables. In other words, during selectivity estimation we are looking for a specific ("virtual") binding to each query variable, whose topic distribution is represented in its corresponding topical random variable (initially unknown) and determined by query predicates. Further, the assumption allows to model the probability $\mathcal{P}(Q)$ in a simple query-specific BN (cf. Fig. 6). Here, the probability of

[8]Note, attribute label a is omitted in the notation, since topic models do not distinguish attributes.

observing a query predicate is solely dependent on the topics of query variables, which enables us to handle dependencies among query predicates in a simplistic manner (see following paragraphs). Note, a generic query-specific BN is given in Fig. 6-a, while Fig. 6-b gives a BN for our running example.

The topical independence assumption leads to a *valid* BN, as the following theorem holds:

Theorem 2. The query-specific BN constructed according to the topical independence assumption is acyclic.

Proof Sketch. BN parts resembling class and string variables form a forest of trees – each tree has depth one. Such trees are combined via relation predicate variables, which have no children (cf. Fig. 6-a). Thus, no cycle can be introduced ■

By means of the query-specific BN, we may compute the query probability $\mathcal{P}(Q)$ as:

$$\mathcal{P}(Q) = \mathcal{P}\Big(\bigwedge \mathbf{X}_w = \mathbf{T} \quad \bigwedge \mathbf{X}_c = \mathbf{T} \quad \bigwedge \mathbf{X}_r = \mathbf{T} \Big) \tag{1a}$$

$$\underset{\text{CR}}{\approx} \prod_{\langle v,a,w\rangle \in Q} \mathcal{P}(X_w \mid X_v) \cdot \prod_{\langle v,type,c\rangle \in Q} \mathcal{P}(X_c \mid X_v) \cdot \prod_{\langle v,r,y\rangle \in Q} \mathcal{P}(X_r \mid X_v, X_y) \tag{1b}$$

with $\mathbf{X}_w$, $\mathbf{X}_c$, $\mathbf{X}_r$ as sets of all string, class, and relation random variables in Q. CR refers to the chain rule, [14]. To compute the above joint probability, we need the topic distributions of topical variables X_v and X_y. However, as these are hidden, we learn their distributions from observed predicate variables, i.e., $\mathbf{X}_w$, $\mathbf{X}_c$, $\mathbf{X}_r$.

We first discuss parameter learning for observed random variables, given topical random variables, and subsequently present learning of hidden topical random variables. It is important to note that *no inferencing is needed* for estimating $\mathcal{P}(Q)$, **Req.4**. This has the positive side-effect that a TRM synopsis may be kept on disk.

Query Predicate Probabilities. Query predicates probabilities are influenced by their associated topical random variables and their probabilities from the TRM synopsis. Thus, we can formulate the conditional probability for X_c, X_r and X_w by incorporating topic distributions of query variables with probabilities estimated using TRM parameters, i.e., $\boldsymbol{\beta}$, $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$. That is, probabilities are obtained as follows:

(1) *Class Predicate Variables.* Adhering to the topical independence assumption, the probability of observing a class, $P(X_c = \mathbf{T})$, is only dependent on its topical variable X_v. We use TRM parameter $\boldsymbol{\lambda}$ to obtain the weight λ_{ct}, indicating the correlation between topic t and class c. The probability of observing class c is given by:

$$P(X_c = \mathbf{T} \mid X_v, \boldsymbol{\lambda}) = \sum_{t \in \mathcal{T}} P(X_v = t) \frac{\lambda_{ct}}{\sum_{t' \in \mathcal{T}} \lambda_{ct'}}$$

Example 5. *Fig. 6-b illustrates two class variables, X_{movie} and X_{person}, which are dependent on the random variables X_m and X_p, respectively. For computing query probabilities, $P(X_{movie} = \mathbf{T})$ and $P(X_{person} = \mathbf{T})$, the corresponding TRM parameters λ_{movie} and λ_{person} (Fig. 5) are used. For instance, given $P(X_m = t_1) = 0.6$, $P(X_m = t_2) = 0.1$, and $P(X_m = t_3) = 0.3$, we have: $P(X_{movie} = \mathbf{T}) = 0.6 \cdot {}^{3}\!/_{4} + 0.1 \cdot {}^{0}\!/_{4} + 0.3 \cdot {}^{1}\!/_{4} = 0.525$.*

(2) *Relation Predicate Variables.* Every relation predicate $\langle v, r, y\rangle$ connects two query variables, for which there are corresponding topical variables X_v and X_y. The variable X_r (representing the relation predicate) solely dependents on the topics of X_v and X_y. The dependency "strength" between r and topics of these two variables is given by TRM parameter ω_r. Using ω_r, the probability of observing relation r is:

$$P(X_r = \mathbf{T} \mid X_v, X_y, \omega_r) = \sum_{t,t' \in \mathcal{T}} \frac{P(X_v = t)\ \omega_{rtt'}\ P(X_y = t')}{\sum_{t''t''' \in \mathcal{T}} \omega_{rt''t'''}}$$

Example 6. *In Fig. 6-b, there are two relation predicate variables: $X_{starring}$ and X_{bornin}. Each of them is dependent on two topical variables, e.g., X_m and X_p condition $X_{starring}$. Probability $P(X_{starring} = \mathbf{T})$ is estimated via matrix $\omega_{starring}$, Fig. 5.*

(3) *String Predicate Variables.* For each string predicate $\langle v, a, w\rangle \in Q$, there is a random variable X_w. The TRM parameter β_{tw} represents the probability of observing word w given topic t. Thus, the probability $P(X_w = \mathbf{T})$ is calculated as probability of observing w, given the topics of its topic variable X_v and $\beta_{1:k}$:

$$P(X_w = \mathbf{T} \mid X_v, \beta_{1:K}) = \sum_{t \in \mathcal{T}} P(X_v = t) \frac{\beta_{tw}}{\sum_{t' \in \mathcal{T}} \beta_{t'w}}$$

Example 7. *Fig. 6-b depicts three string predicate variables, needed for the string predicates comprised in our query (Fig. 2). Given $P(X_m)$ as in the example above, the probability of observing "holiday", $P(X_{holiday} = \mathbf{T})$, is:*

$$P(X_{holiday} = \mathbf{T}) = 0.6 \cdot \frac{0.011}{0.017} + 0.1 \cdot \frac{0.002}{0.017} + 0.3 \cdot \frac{0.004}{0.017} = 0.47$$

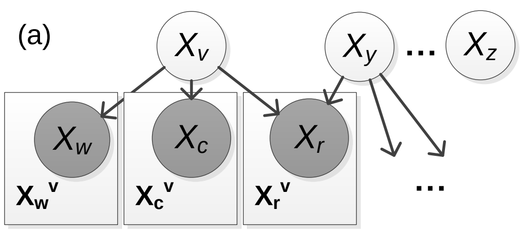
Learning Topics of Query Variables. The core idea of TopGuess is to find an optimal topic distribution for every topic variable, so that the joint probability of the query-specific BN is maximized, Eq. 1. Thus, as a final step, we learn parameters for our initially unobserved topical random variables, based on observed predicate variables. For computing these parameters, we first introduce a set of topic parameters θ_{vt} for each topical random variable X_v. $\boldsymbol{\theta} = \{\theta_{vt} | v \in V_{Q_V}, t \in \mathcal{T}\}$ denotes the set of parameters for all topical variables. As before, $\mathbf{X}_w$, $\mathbf{X}_c$ and $\mathbf{X}_r$ denote string, class and relation predicate variables in a query-specific BN. Then, we find parameters $\boldsymbol{\theta}$ for topic variables, which maximize the log-likelihood of Eq. 1. The optimization problem is:

$$\arg\max_{\boldsymbol{\theta}} \ \ell(\boldsymbol{\theta} : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r)$$

where $\ell(\boldsymbol{\theta} : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r)$ is the log-likelihood defined as:

$$\begin{aligned} \ell(\boldsymbol{\theta} : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r) &= P(\mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r | \boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\omega}, \boldsymbol{\lambda}) \\ &= \sum_{v} \sum_{X_w \in \mathbf{X}_w^v} \log P(X_w | X_v, \boldsymbol{\beta}) \\ &+ \sum_{v} \sum_{X_c \in \mathbf{X}_c^v} \log P(X_c | X_v, \boldsymbol{\lambda}) \\ &+ \sum_{v,y} \sum_{X_r \in \mathbf{X}_r^{v,y}} \log P(X_r | X_v, X_y, \boldsymbol{\omega}) \end{aligned}$$

where $\mathbf{X}_w^v$ and $\mathbf{X}_c^v$ is the set of all string and class random variables with a parent X_v. $\mathbf{X}_r^{v,y}$ is the set of all relation random variables with parents X_v and X_y. We use gradient ascent



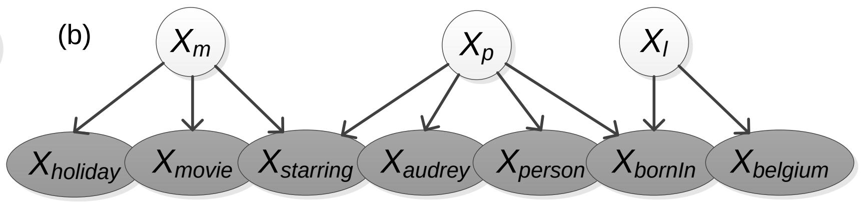


Fig. 6. (a) Generic query-specific BN in plate notation. Notice, string predicate variables (X_w), class predicates variables (X_c), and relation predicate variables (X_r) are only dependent on their topical random variable X_v, X_y. (b) A query-specific BN for query in Fig. 2 with 3 topical variables (e.g., X_m), 2 class predicate variables (e.g., X_{movie}), 2 relation predicate variables (e.g., $X_{starring}$) and 3 string predicate variables (e.g., $X_{holiday}$). Observed variables (dark Grey) are independent from each other and only dependent on hidden topical random variables (light Grey) (cf. topical ind. asmp., Def. 3).

optimization to learn the parameters. First, we parametrize each $P(X_v = t)$ with θ_{vt} as $P(X_v = t) = \frac{e^{\theta_{vt}}}{\sum_{t' \in \mathcal{T}} e^{\theta_{vt'}}}$ to obtain a proper probability distribution over the topics. Obtaining the gradient requires dealing with the log of the sum over the topics of each topical variable. Therefore, we make use of theorem [14]:

Theorem 3. Given a BN and $\mathcal{D} = \{\mathbf{o}[1], \dots, \mathbf{o}[M]\}$ as a partially observed dataset. Further, let X be a random variable in that BN, and let $\mathbf{Pa}(X)$ denote its parents. Then:

$$\frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial P(x|\mathbf{pa})} = \frac{1}{P(x|\mathbf{pa})} \sum_{m=1}^{M} P(x, \mathbf{pa}|\mathbf{o}[m], \boldsymbol{\theta}),$$

This provides the form of the gradient needed. Now, the gradient of the log-likelihood w.r.t. parameter θ_{vt} is:

$$\frac{\partial \ell(\boldsymbol{\theta} : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r)}{\partial \theta_{vt}} = \frac{\partial \ell(\boldsymbol{\theta} : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r)}{\partial P(X_v = t)} \frac{\partial P(X_v = t)}{\partial \theta_{vt}}$$

The first part of the gradient is obtained via Theorem 3:

$$\frac{\partial \ell(\boldsymbol{\theta} : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r)}{\partial P(X_v = t)} = \frac{1}{P(X_v = t)} \cdot \left(\sum_{X_w \in \mathbf{X}_w^v} P(X_v = t|X_w, \boldsymbol{\beta}) + \sum_{X_c \in \mathbf{X}_w^v} P(X_v = t|X_c, \boldsymbol{\lambda}) + \sum_{y} \sum_{X_r \in \mathbf{X}_r^{v,y}} P(X_v = t|X_r, X_y, \boldsymbol{\omega}) \right)$$

Using the Bayes rule we have:

$$\frac{\partial \ell(\boldsymbol{\theta} : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r)}{\partial P(X_v = t)} = \sum_{X_w \in \mathbf{X}_w^v} \frac{P(X_v = t) P(X_w|\boldsymbol{\beta}, t)}{\sum_{t'} P(X_v = t') P(X_w|\boldsymbol{\beta}, t')} + \sum_{X_c \in \mathbf{X}_w^v} \frac{P(X_v = t) P(X_c|\boldsymbol{\lambda}, t)}{\sum_{t'} P(X_v = t') P(X_c|\boldsymbol{\lambda}, t')} + \sum_{y} \sum_{X_r \in \mathbf{X}_r^{v,y}} \frac{P(X_v = t) \sum_{t'} P(X_r|X_y, \boldsymbol{\omega}, t')}{\sum_{t''} P(X_v = t'') \sum_{t'''} P(X_r|X_y, \boldsymbol{\omega}, t''')}$$

Finally, the second part of the gradient is given by:

$$\frac{\partial P(X_v = t)}{\partial \theta_{tv}} = \frac{e^{\theta_{tv}} \sum_{t'-t} e^{\theta_{t'v}}}{(\sum_{t'} e^{\theta_{t'v}})^2}$$

Estimation Complexity. We give a complexity bound for query probability estimation, Eq. 1, as:

Theorem 4 (Time Complexity of $\mathcal{P}(Q)$)**.** Given k topics and a query Q, the time for computing $\mathcal{P}(Q)$ is in $O(\psi \cdot |Q| \cdot k)$, with ψ as number of iterations needed for optimization.

Proof. Complexity for $\mathcal{P}(Q)$ is comprised of (1) estimation time for the joint probability of Q's query-specific BN, and (2) time necessary for learning optimal topic distributions. Given topic distributions for each X_v, the former step requires only a simple summing out of the variables $\mathbf{X}_v$. Thus, its time is $\in O(|Q| \cdot K)$, with $|Q|$ and K as number of query predicates and topics, respectively. For the latter step, let an optimization algorithm require ψ iterations to reach an optimum. Note, ψ is a constant only driven by the error threshold of the optimization problem, thus, independent of $|Q|$, K or synopsis size $\mathcal{S}$. For each such iteration we require an update of variables $\mathbf{X}_w$, $\mathbf{X}_c$, and $\mathbf{X}_r$, as well as topic model parameter $\boldsymbol{\theta}$. Note, while the number of random variables $\mathbf{X}_i$, $i \in \{w, c, r\}$, is bounded by $|Q|$, $\boldsymbol{\theta}$ is bound by K. Thus, we update $O(K \cdot |Q|)$ values – each in constant time, $O(1)$. Overall, the second task requires a complexity of $O(\psi \cdot K \cdot |Q|)$. Therefore, step (1) and (2) combined take $O(\psi \cdot K \cdot |Q|)$ time ∎

Note, ψ is determined by the specific algorithm used for optimization. We use a gradient ascent approach, which is known to have a tight bound of iterations $\in O(\epsilon^2)$, with an arbitrarily small $\epsilon > 0$ [21]. Overall, complexity for $\mathcal{P}(Q)$ is *independent of the synopsis size* $\mathcal{S}$ (**Req.5**).

V. Evaluation

We conducted experiments to analyze the effectiveness (**Req.1+3**) and the efficiency (**Req.2+4+5**) of selectivity estimation using TopGuess. By means of the former, we wish to compare the quality of estimates. Previous work has shown that estimation quality is of great importance for many use cases, most notably for query optimization [6]. The latter inspects the applicability towards real-world systems.

Overall, our results are very promising. In terms of effectiveness, we could gain up to 93% more accuracy by TopGuess. While TopGuess stored a very fine-grained synopsis on disk and relied on a simplistic topic learning procedure, it still achieved similar runtime performances as the baselines. In fact, we expect drastic improvements can be done as future work. In general, we noted that TopGuess runtime behavior was much less influenced by query respectively synopsis size than for baseline approaches. Further, memory consumption was negligible for TopGuess, in contrast to the baselines.

A. Evaluation Setting

Systems. We compare TopGuess with two kinds of baselines. First, string predicates are combined with structured predicates via an *independence assumption*, `ind`. More precisely, the selectivity of string and structured predicates is estimated using a string synopsis and histograms based on [22], respectively. Obtained probabilities were combined in a greedy fashion assuming independence.

Second, multiple query predicates are combined relying on a BN, `bn`. That is, we reuse our work on PRMs for text-rich data graphs [7]. See Ex. 3 for such a PRM. `bn` handles the problem of multiple value assignments to a single random variable via aggregation functions. We use a *stochastic mode* aggregation, which essentially uses all assignments, but weights each one with its frequency [16].

As string synopsis function ν we exploit n-gram synopses [10]. Such a synopsis reduces a vocabulary by using a predefined criterion to dictate which n-grams to in-/exclude. A simplistic strategy is to choose random n-gram *samples* from the data. Another approach is to construct a *top-k* n-gram synopsis. For this, n-grams are extracted from the data together with their counts. Then, the k most frequent n-grams are included in the synopsis. Further, a *stratified bloom filter* synopsis has been proposed [10], which uses bloom filters as a heuristic map that projects n-grams to their counts. We use these three types of synopses in our experiments. Thus, a synopsis represents a subset of all possible n-grams occurring in the data. Note, we refer to excluded n-grams as "missing". The probability for missing n-grams cannot be estimated with a probabilistic framework, as such strings are not included in a sample space. To deal with this case, a string predicate featuring a missing n-gram is assumed to be independent from the remainder of the query. Then, its probability can be estimated based on various heuristics. We employ the *leftbackoff* strategy, which finds the longest known n-gram that is the pre- or postfix of the missing keyword and estimates its probability based on the statistics for that pre- and postfix [10].

Combining string synopses with our two categories of baselines results in six different systems: $\texttt{ind}_{\text{sample}}$, $\texttt{ind}_{\text{top-}k}$ and $\texttt{ind}_{\text{sbf}}$ rely on the independence assumption, $\texttt{bn}_{\text{sample}}$, $\texttt{bn}_{\text{top-}k}$ and $\texttt{bn}_{\text{sbf}}$ represent BN approaches.

Data. We employ two real-world RDF datasets for evaluation: DBLP[9] and IMDB [23]. For both datasets, we could extract large vocabularies containing $25,540,172$ and $7,841,347$ 1-grams from DBLP and IMDB, respectively. See also Table I for an overview. Notice, while IMDB as well as DBLP both feature text-rich attributes like `name`, `label` or `info`, they differ in their overall amount of text: IMDB comprises large texts, e.g., associated via `info`, DBLP, however, holds much less text. On average an attribute in DBLP contains only 2.6 1-grams with a variance of 2.1, in contrast to IMDB with 5.1 1-grams, given a variance 95.6. Further, also the attribute with the most text associated is larger, having 28.3 1-grams, for IMDB, than for DBLP with 8.1 1-grams (cf. Table I).

Our hypothesis is that these differences will be reflected in different degrees of correlations between text and structured data. Moreover, we are interested in comparing performance of `bn` and TopGuess w.r.t. varying amounts of texts. In particular, as TopGuess uses a topic model synopsis, we wish to analyze its effectiveness in such settings.

Queries. As queries we reuse existing work on keyword

TABLE I
DATASET STATISTICS

	IMDB	DBLP
# Triples	$7,310,190$	$11,014,618$
# Resources	$1,673,097$	$2,395,467$
# Total 1-grams	$7,841,347$	$25,540,172$
# Avg. 1-grams Mean	5.1	2.6
# Avg. 1-grams Variance	95.6	2.1
Max. attr. # avg. 1-grams	28.3	8.1
# Attributes	10	20
# Relations	8	18
# Classes	6	18

TABLE II
STORAGE SPACE (MBYTE)

	IMDB		DBLP	
	Data			
Disk	1600		5800	
	Data Synopsis			
	`bn` & `ind`	TopGuess	`bn` & `ind`	TopGuess
Mem.	$\{2,4,20,40\}$	≤ 0.1	$\{2,4,20,40\}$	≤ 0.1
Disk	0	281.7	0	229.9

search evaluation [23], [24]. We form queries adhering to our model by constructing graph patterns, comprising string, class, and relation predicates that correspond to the given query keywords and their structured results. We generated 54 DBLP queries based on "seed" queries reported in [24]. That is, for each query in [24], we replaced its keyword constants with variables, evaluated such seed queries, and generated new queries by replacing a keyword variable with one of its randomly selected bindings. Additionally, 46 queries were constructed for IMDB based on queries taken from [23]. We omitted 4 queries in [23], as they could not be translated to conjunctive queries. Overall, our load features queries with $[2,11]$ predicates in total: $[0,4]$ relation, $[1,7]$ string, and $[1,4]$ class predicates. As our query model allows solely single keywords to be used, we treat string predicates with phrases as several predicates. Query statistics and a complete query listing are given in the appendix, Sect. VIII.

As hypothesis, we expect queries with a larger number of predicates to be more "difficult" in terms effectiveness and efficiency. Further, we expect correlations between query predicates to have a strong influence on effectiveness. Note, we observe during structure learning of the `bn` baseline systems different degrees of correlations in DBLP and IMDB. More precisely, we noticed that there are more correlated predicates in IMDB, e.g., `name` (class `Actor`) and `title` (class `Movie`), than in DBLP.

Synopsis Size. Using the same configurations as in [7], we employ different synopsis sizes for our baselines `ind` and `bn`. The factor driving the overall synopses size for `ind` and `bn` is their string synopsis size, i.e., the size of their common representation space $|\mathcal{C}|$. This effect is due to $\mathcal{C}$ determining the size of the (conditional) probability distribution in $\texttt{ind}_*$ ($\texttt{bn}_*$). CPDs are very costly in terms of space, while other statistics, e.g., the BN structure, are negligible. We varied the number of 1-grams captured by the top-k and sample synopsis $\in \{0.5K, 1K, 5K, 10K\}$. For the sbf string synopsis, we captured up to $\{2.5K, 5K, 25K, 50K\}$ of the most frequent 1-grams for each attribute and varied the bloom filter sizes, resulting in similar memory requirements. Note, sbf systems featured all 1-grams occurring in our query load.

Except for TopGuess, all systems load the synopsis into

[9]http://knoesis.wright.edu/library/ontologies/swetodblp/

main memory. To be more precise, only $\texttt{bn}_*$ approaches require the synopsis to be in-memory for inferencing. For comparison, however, we also load statistics for $\texttt{ind}_*$ approaches. Different string synopses (sizes) translate to approaches consuming $\{2, 4, 20, 40\}$ MByte of memory, Table II. In contrast to $\texttt{bn}_*$ and $\texttt{ind}_*$, TopGuess keeps a large topic model at disk, and constructs a small, query-specific BN in memory at runtime (memory consumption $\leq$ 100 KBytes on average). Thus, all query-independent statistics remain on the hard-disk. The large disk size $\in [220 - 280]$ MByte for TopGuess comes from the use of all 1-grams in the data. Table II shows an overview of the main memory and disk usage required by the different systems.

We expect TopGuess's fine-grained model to enable very accurate estimations. At the same time, such a disk-based, extensive synopsis may come with runtime problems. That is, in the following we aim to empirically validate our estimation time complexity bounds.

Implementation and Offline Learning. For baselines $\texttt{bn}_*$ and $\texttt{ind}_*$, we started by constructing string synopses. Each synopsis, including sbf-based synopses, was learned in $\leq$ 1h. As $\texttt{bn}_*$ and $\texttt{ind}_*$ use the same probability distributions (BN parameters), parameters were trained together. For $\texttt{bn}_*$ we use a PRM construction as done in [7]. That is, we capture un-/structured data elements using random variables and learn correlations between them, thereby forming a network structure. For efficient selectivity estimation the network is reduced to a "lightweight" model, capturing solely the most important correlations. Then, we calculate model parameters (CPDs) based on frequency counts. For $\texttt{ind}_*$ systems, we do not need the model structure and merely keep the marginalized parameters. Structure and parameter learning combined took in the worst case up to three hours. The structure and the parameters are stored in a key-value store outside the database system – both were loaded at start-up. Depending on the synopsis size, loading the model into memory took up to $3s$. The inferencing needed by the $\texttt{bn}_*$ systems for selectivity estimation is done using a Junction tree algorithm [7].

TopGuess exploits an "off-the-shelf" TRM from [13]. That is, standard TRM learning was employed over a data sample. We sampled up to 30K entities per class from each dataset, to ensure that all classes and relations are equally represented. The number of TRM topics is an important factor, determining which correlations are discovered. Thus, a sufficient number of topics is needed to correlate text with a heterogeneous set of classes and predicates. Note, a TRM is a supervised topic model, which handles the sparseness of these topics correlated to structure information. In other words, some topics can be correlated with many structure elements, whereas others are not. We experimented with a varying number of topics $\in [10 - 100]$ and found that, for datasets employed in our evaluation, 50 topics are enough to capture all important correlations. Notice, a TRM may easily be extended to determine the number of topics based on the data in an unsupervised fashion, by using a non-parametric Bayesian model [18]. The TRM learning took up to 5h, and the resulting models were stored in an inverted index on hard disk, Table II. At query time, we employed a greedy gradient ascent algorithm for learning the topic variable distributions. To avoid local maxima, we used up to 10 random restarts.

We implemented all systems in Java 6. Experiments were run on a Linux server with two Intel Xeon CPUs at 2.33GHz, 48 GB RAM (with JVM using 16 GB), and a RAID10 with IBM SAS 10k rpm disks. Before each query execution, OS caches were cleared. Values are averages over five runs.

B. Selectivity Estimation Effectiveness

We employ the *multiplicative error* metric (me) [25] for measuring effectiveness:

$$me(Q) = \frac{\max(sel(Q), \overline{sel}(Q))}{\min(sel(Q), \overline{sel}(Q))}$$

with $sel(Q)$ and $\overline{sel}(Q)$ as exact and approximated selectivity for Q, respectively. Intuitively, me gives the factor to which $\overline{sel}(Q)$ under- or overestimates $sel(Q)$.

Overall Results. Fig. 7-a, -b (-e, -f) show the multiplicative error vs. synopsis size (number of predicates) for DBLP and IMDB, respectively. As expected, baseline system effectiveness strongly depends on the synopsis size. That is, for small synopses $\leq$ 20 MByte, $\texttt{ind}_*$ and $\texttt{bn}_*$ performed poorly. We explain this with the information loss, due to omitted 1-grams in the common representation space (**Req.3**). That is, the employed string synopsis traded space for accuracy, and heuristics had to used for probability estimation. In fact, in case of $\texttt{bn}_*$, the information loss is aggravated as missed keyword can not be added to an unrolled BN, instead one must assume independence between such a string predicate and the remainder of the query.

TopGuess, on the other hand, did not suffer from this issue, as all query-independent statistics could be stored at disk, and solely the query-specific BN was loaded at runtime. Thus, TopGuess could exploit very fine-grained probabilities, and omitted any kind of heuristics. We observed that, on average, TopGuess could reduce the error of the best performing $\texttt{bn}_{\text{sbf}}$ by 93% (33%), and the best system using the independence assumption, $\texttt{ind}_{\text{sbf}}$, by 99% (35%) on IMDB (DBLP). Further, we noted TopGuess to be less driven by the overall data correlations. That is, while $\texttt{bn}$ systems were strongly affected by the little correlations in DBLP, TopGuess outperformed the best baseline by 33%. Overall, this confirms our initial expectation that the query-specific BN and the TRM synopsis are well-suited for modeling fine query/data dependencies.

As in previous work [7], different string synopses in $\texttt{ind}_*$ and $\texttt{bn}_*$, yielded different estimation effectiveness results. Sampling-based systems were outperformed by systems using top-k n-grams synopses, which in turn, performed worse than sbf-based approaches. These drastic misestimates come from query keywords being "missed" in the string synopsis. Thus, we can see estimation quality being strongly influenced by accurate string probabilities. We argue that such results clearly show the need for a data synopsis differing from current approaches – allowing for on-disk storage of statistics, **Req.2**.

Synopsis Size. Fig. 7-a/-e shows estimation errors w.r.t. in-memory synopsis size. An important observation is that synopsis size is a key factor for estimation effectiveness. Top-k and especially sample-based n-gram systems were strongly affected by the (string) synopsis size. With increasing size, more relevant query keywords were capture in the data synopsis, leading to less inaccurate heuristics being applied. Further, we noted performance of sbf-based approaches to be

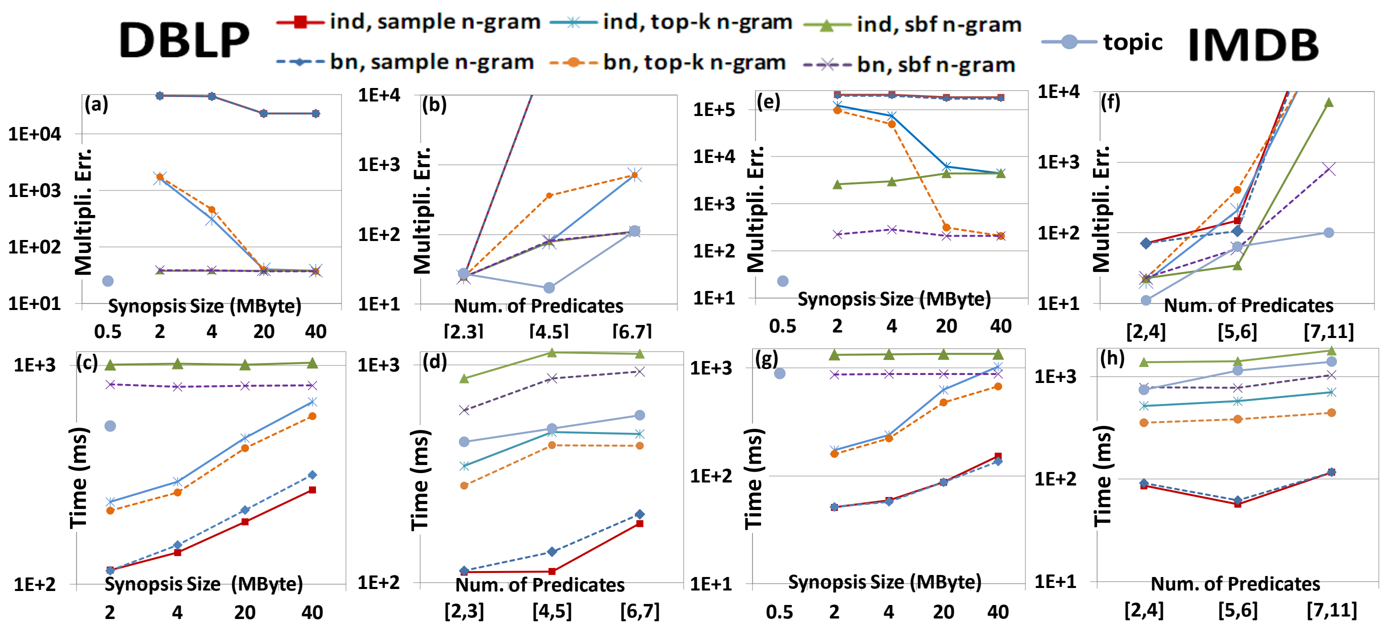


Fig. 7. Effectiveness: (a)+(b) for DBLP, and (e)+(f) for IMDB. Efficiency: (c)+(d) for DBLP, and (g)+(h) for IMDB. Y-axes are given in logarithmic scale.

fairly stable. This is due to sbf being able to capture all query keywords, thus, no heuristics were necessary. However, while sbf-based approaches proofed to be an effective strategy, they also inherently suffer from limited (in-memory) space, and thus must eventually discard words in the vocabulary.

In contrast, TopGuess memory usage is a small constant $\leq$ 0.1 MByte. Yet, TopGuess resolves the issue of missing n-grams completely: the query-independent TRM stored on disk captures statistics for all n-grams. At runtime, TopGuess retrieves the necessary n-grams for a particular query, and constructs its query-specific BN (**Req.3**). This extremely compact BN can be explained with TopGuess only using random variables that either are *binary* or have a sample space bound by the number of topics, k. Note, both such factors are constant in the overall data synopsis size, **Req.4**.

Correlations and Amount of Text. Following our initial hypothesis, we found performances for IMDB and DBLP to vary. Given IMDB, TopGuess and $\texttt{bn}_{\text{sbf}}$ could reduce errors of the $\texttt{ind}_{\text{sbf}}$ approach by 93% and 99%, respectively. On the other hand, for DBLP improvements were much smaller. These differences are due to the varying degree of correlations in our two datasets. While learning the BNs for `bn`, we observed significantly less correlations in DBLP than in IMDB. More importantly, many of the DBLP queries include string predicates such as `name` and `label`, for which we could not observe any correlations. For such queries, the probabilities obtained by $\texttt{bn}_*$ were close to the ones computed by $\texttt{ind}_*$.

Further, we noted that the degree of correlation between un-/structured data, is greatly influenced by the number/length of texts. Essentially, we noted that given attributes with more text values, more correlations among them and/or structured data tend to occur. For instance, given queries involving attribute `name` in DBLP, with only 2.4 1-grams (variance: 2.1 1-grams, cf. Table I), with measured over 30% less correlations than for queries on IMDB with attribute `info`.

However, compared to $\texttt{bn}_*$, TopGuess relies on a fine-grained BN: while $\texttt{bn}_*$ exploits correlations observed in the data graph before query time, TopGuess utilizes all query predicate correlations via a query-specific BN at runtime. Thus, a TopGuess BN is able to capture even "minor" correlations, which may have been discarded by $\texttt{bn}_*$ in favor of a compact structure. Note that previous works on PRMs for selectivity estimation [6], [7], aim at a "lightweight" model structure, i.e., dependency informations is traded for efficient storage and inferencing. Such trade-offs, however, are not necessary for TopGuess. Thus, even for the less correlated dataset DBLP, TopGuess outperforms the baseline $\texttt{ind}_{sbf}$ and $\texttt{bn}_{sbf}$ by 35% and 33%. We argue that this result also confirms the general applicability of TopGuess. Even for "little" text data, TopGuess was able to capture meaningful topics, leading to accurate probability estimates for query-specific BNs, **Req.1**.

Query Size. In Fig. 7-b/-f we depict multiplicative error (average over synopsis sizes) vs. number of query predicates. As expected, estimation errors increase for all systems in the number of query predicates. For our baselines, we explain this behavior via: (1) given a higher number of predicates chances of "missing" a keyword increase, and (2) when missing an n-grams, the error is propagated to the estimate for the remainder of the query (which might have been fine otherwise). However, while the TopGuess approach also led to more misestimates for larger queries, the degree of this increase was smaller. In particular, considering highly correlated queries for IMDB with size $\in [7-11]$, we can observe (Fig. 7 -f) TopGuess to perform much more stable than $\texttt{bn}_*$ or $\texttt{ind}_*$.

As in [7], we also noticed misestimates of `bn` due to inaccurate stochastic value aggregation. This effect led to $\texttt{ind}_*$ outperforming $\texttt{bn}_*$ for some queries. TopGuess does not suffer from such a problem, because its random variables are "predicate-specific", i.e., we construct one single random variable for each query predicate at runtime, **Req.1**. Overall, compared to $\texttt{bn}_*$ respectively $\texttt{ind}_*$, TopGuess yielded the most accurate and stable performance.

C. Selectivity Estimation Efficiency

We now analyze the estimation efficiency for varying synopses sizes, Fig. 7-c/-g, and query complexities, Fig. 7-d/-h. For TopGuess, its estimation times comprise loading and BN construction as well as topic learning. For `bn` and `ind`, the reported times represent solely the inference task, i.e., time for model construction and loading have been omitted.

Overall Results. As noted in [7], we also observed that for `bn`/`ind` not BN inferencing, but the string synopsis was driving the performance. Intuitively, the more n-gram were missed, the "simpler" and the more efficient these systems became. However, such performance gains come at the expense of estimation effectiveness – the fastest baseline system relied on sample-based synopses. In fact, the very same systems performed worst in terms of effectiveness. This strengthens our believe that state-of-the-art systems exhibit a strong trade-off between estimation time and quality – contradicting **Req.1**.

Comparing the two systems with best effectiveness, i.e., TopGuess and $\texttt{bn}_{\text{sbf}}$, TopGuess led to a better performance by 29%. However, in comparison to top-k systems, TopGuess resulted in a performance decrease of 28%. We explain these performance drawbacks with the time-consuming disk I/O, which was needed for loading the necessary statistics.

However, TopGuess performance results are still promising: (1) Its efficiency it is not driven by the overall synopsis size. That is, while `bn` and `ind` clearly outperform TopGuess, given small synopses ≤ 4 MByte, TopGuess results are better respectively comparable for synopses ≥ 20 MByte. We expect such effects to be even more drastic for "very large" `bn` (`ind`) synopses $\gg 100$ MByte. (2) Memory space consumption was very low, ≤ 1 MByte, for TopGuess. This is a clear benefit for systems with limited resources. (3) As we will discuss below, we also found that TopGuess performance was much less driven by query size. Considering both aspects, TopGuess guarantees a much more "stable" behavior, **Req.5**.

Synopsis Size. Fig. 7-c/-g shows selectivity estimation time vs. synopsis size. For baseline systems we can see a strong dependency between synopsis size and their runtime behavior: **Req.5** fails. While `bn` and `ind` reach high efficiency for synopses ≤ 4 MByte, there performance decreases rapidly with synopses ≥ 20 MByte. Note, sbf-based approaches, are an exception, as there computational costs are determined by bloom filters and not their overall number of 1-grams. TopGuess, does not suffer from this issue. As our approach does not require any marginalization or inferencing, constructing a query-specific BN and computing its joint probability is *independent from the size of the TRM*, **Req.5**.

Regarding memory consumption, we observed drastic differences when comparing `bn` or `ind` with TopGuess. Our system required overall ≤ 1 MByte of memory, as only its query-specific BN was loaded during runtime. Every other statistic was kept on hard disk. Further, in contrast to an unrolled BN, TopGuess's BN only features binary random variables and variables with topics as sample space. Baselines, however, inherently needed much more space. For instance, capturing $5K$ 1-grams per attribute resulted in ≈ 20 MByte. This is because their random variables had actual 1-grams as sample space. Note, `ind` systems may actually be also kept on-disk. We loaded their statistics for comparison with `bn`.

Query Size. For all systems estimation times increase with query size, cf. Fig. 7-d/-h. However, as TopGuess exploits an extremely compact query-specific BN, we expected it's performance to be much less influenced by query size. To confirm this we compared the *standard deviation* of the estimation time between TopGuess and $\texttt{bn}_*$ w.r.t. different sizes. The standard deviations was $82,48$ ms and $213,48$ ms for TopGuess and $\texttt{bn}_*$, respectively. The low deviation for TopGuess indicates that the required I/O and probability estimations times varied little w.r.t. query size. For $\texttt{bn}_*$, however, its high variance suggests that the performance is strongly affected.

VI. Related Work

For selectivity estimation on structured data, works exploit *table-level* data synopses, which capture attributes within the same table, e.g., [2]. Other approaches focus on *schema-level* synopses, which are not restricted to a single table, but capture attributes and relations: graph synopses [4], join samples [3], and graphical models [5], [6], [7].

In contrast to TopGuess, such approaches do not summarize correlations in text. In fact, in [7] we loosely integrated BN and string synopses. However, [7] does not provide an uniform framework. Further, it suffers from the same problems as previous PRM-based solution [5], [6]. We discussed their drawbacks in depth throughout the paper, Sect. III + V.

For estimating string predicates selectivity, language models and other machine learning techniques have been utilized [8], [9], [26], [10]. Some works aim at substring or fuzzy string matching [8], [26], while other approaches target "extraction" operators, e.g., dictionary-based operators [10].

However, these approaches do not consider dependencies among various string predicates and/or with query predicates for structured data. In contrast, we present a *holistic approach* for hybrid queries.

VII. Conclusion

We provided a framework for selectivity estimation on hybrid queries, and analyzed state-of-the-art solutions. Driven by our findings, we proposed a novel, holistic estimation of hybrid queries, TopGuess. We gave space and time complexity bounds for TopGuess, by means of a theoretical analysis. We conducted extensive empirical studies on real-world data. TopGuess achieved strong effectiveness improvements, while not requiring additional runtime. We are confident, however, that TopGuess may allow for runtime improvements in the future, using sophisticated topic learning or caching.

References

[1] A. Maier and D. E. Simmen, "Db2 optimization in support of full text search," *IEEE Data Eng. Bull.*, vol. 24, no. 4, pp. 3–6, 2001.

[2] V. Poosala, P. Haas, Y. Ioannidis, and E. Shekita, "Improved histograms for selectivity estimation of range predicates," *SIGMOD*, 1996.

[3] S. Acharya, P. Gibbons, V. Poosala, and S. Ramaswamy, "Join synopses for approximate query answering," in *SIGMOD*, 1999.

[4] J. Spiegel and N. Polyzotis, "Graph-based synopses for relational selectivity estimation," in *SIGMOD*, 2006.

[5] L. Getoor, B. Taskar, and D. Koller, "Selectivity estimation using probabilistic models," in *SIGMOD*, 2001.

[6] K. Tzoumas, A. Deshpande, and C. S. Jensen, "Lightweight graphical models for selectivity estimation without independence assumptions," *PVLDB*, 2011.

[7] A. Wagner, V. Bicer, and T. D. Tran, "Selectivity estimation for hybrid queries over text-rich data graphs," in *EDBT*, 2013.

[8] S. Chaudhuri, V. Ganti, and L. Gravano, "Selectivity estimation for string predicates: Overcoming the underestimation problem," in *ICDE*, 2004.

[9] L. Jin and C. Li, "Selectivity estimation for fuzzy string predicates in large data sets," in *VLDB*, 2005.

[10] D. Z. Wang, L. Wei, Y. Li, F. Reiss, and S. Vaithyanathan, "Selectivity estimation for extraction operators over text data," in *ICDE*, 2011.

[11] J. Chang and D. Blei, "Relational topic models for document networks," in *AIStats*, 2009.

[12] J. Zeng, W. K. Cheung, C.-h. Li, and J. Liu, "Multirelational topic models," in *ICDM*, 2009.

[13] V. Bicer, T. Tran, Y. Ma, and R. Studer, "Trm - learning dependencies between text and structure with topical relational models," in *ISWC*, 2013.
[14] D. Koller and N. Friedman, *Probabilistic graphical models*. MIT press, 2009.
[15] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *JMLR*, vol. 3, pp. 993–1022, 2003.
[16] B. Taskar, E. Segal, and D. Koller, "Probabilistic classification and clustering in relational data," in *IJCAI*, 2001.
[17] G. F. Cooper, "The computational complexity of probabilistic inference using bayesian belief networks," *Artif. Intell.*, vol. 42, no. 2-3, pp. 393–405, 1990.
[18] F. Doshi, K. Miller, J. V. Gael, and Y. W. Teh, "Variational inference for the indian buffet process," *JMLR*, vol. 5, pp. 137–144, 2009.
[19] A. Smola and S. Narayanamurthy, "An architecture for parallel topic models," *VLDB*, 2010.
[20] A. Banerjee and S. Basu, "Topic models over text streams: A study of batch and online unsupervised learning," in *SDM*, 2007.
[21] C. Cartis, N. I. M. Gould, and P. L. Toint, "On the complexity of steepest descent, newton's and regularized newton's methods for nonconvex unconstrained optimization problems," *SIAM*, vol. 20, no. 6, pp. 2833–2852, 2010.
[22] H. Huang and C. Liu, "Estimating selectivity for joined rdf triple patterns," in *CIKM*, 2011.
[23] J. Coffman and A. C. Weaver, "A framework for evaluating database keyword search strategies," in *CIKM*, 2010.
[24] Y. Luo, W. Wang, X. Lin, X. Zhou, J. Wang, and K. Li, "Spark2: Top-k keyword query in relational databases," *TKDE*, vol. 23, no. 12, pp. 1763–1780, 2011.
[25] A. Deshpande, M. N. Garofalakis, and R. Rastogi, "Independence is good: Dependency-based histogram synopses for high-dimensional data," in *SIGMOD*, 2001.
[26] H. Lee, R. T. Ng, and K. Shim, "Extending q-grams to estimate selectivity of string matching with low edit distance," in *VLDB*, 2007.

VIII. Appendix

Below, we present statistics as well as a complete listing of queries used during our experiments. Note, queries for the DBLP dataset are based on [24], while IMDB queries are taken from [23]. All queries are given in RDF N3[10] notation.

TABLE III
Query Statistics

Predicates:	#Relation			#String		
	0	1	[2, 4]	[1, 2]	3	[4, 7]
# Queries	33	44	23	28	35	26
Predicates:	#Class			#Total		
	1	2	[3, 4]	[2, 3]	[4, 6]	[7, 11]
# Queries	49	30	21	28	31	41

Listing 1. Queries for DBLP [24]

```
# @prefix dc:
# http://purl.org/dc/elements/1.1/> .
# @prefix foaf:
# <http://xmlns.com/foaf/0.1/> .
# @prefix rdf:
# <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
# @prefix rdfs:
# <http://www.w3.org/2000/01/rdf-schema#> .
# @prefix dblp:
# <http://lsdis.cs.uga.edu/projects/semdis/opus#>

# q1
?x rdfs:label "clique" .
?x dblp:last_modified_date "2002-12-09" .
?x rdf:type dblp:Article_in_Proceedings .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "nikos" .

# q2
?y rdf:type foaf:Person .
?y foaf:name "nikos" .
?y foaf:name "zotos" .

# q3
?x rdfs:label "constraint" .
?x dblp:last_modified_date "2005-02-25" .
?x rdf:type dblp:Article_in_Proceedings .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "chuang" .

# q4
?x rdfs:label "mining" .
?x rdfs:label "clustering" .
?x dblp:year "2005" .
?x rdf:type dblp:Article .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "nikos" .

# q5
?x rdfs:label "spatial" .
?x dblp:last_modified_date "2006-03-31" .
?x rdf:type dblp:Article_in_Proceedings .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "patel" .

# q6
?x rdf:type dblp:Article .
?x rdfs:label "middleware" .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "zhang" .

# q7
?x rdf:type dblp:Article_in_Proceedings .
?x rdfs:label "middleware" .
?x rdfs:label "optimal" .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "ronald" .

# q8
?x rdf:type dblp:Article_in_Proceedings .
?x rdfs:label "partition" .
?x rdfs:label "relational" .
?x rdfs:label "query" .

# q9
?x rdf:type dblp:Article_in_Proceedings .
?x rdfs:label "partition" .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "patel" .

# q10
?x rdf:type dblp:Proceedings .
?x rdfs:label "recognition" .
?x rdfs:label "speech" .
?x rdfs:label "software" .
?x dc:publisher ?p .

# q11
?x rdf:type dblp:Proceedings .
```

[10]http://www.w3.org/TeamSubmission/n3/

```
?x rdfs:label "data" .
?x rdfs:label "mining" .
?x dc:publisher <http://www.springer.de/> .

# q12
?x rdf:type dblp:Proceedings .
?x rdfs:label "australia" .
?x rdfs:label "stream" .
?x dc:publisher <http://www.springer.de/> .

# q13
?x dblp:year "2002" .
?x rdf:type dblp:Proceedings .
?x rdfs:label "industrial" .
?x rdfs:label "database" .
?x dc:publisher ?p .

# q14
?x rdf:type dblp:Article_in_Proceedings .
?x dblp:last_modified_date "2006-03-09" .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "jignesh" .

# q15
?x rdf:type dblp:Article_in_Proceedings .
?x rdfs:label "algorithm" .
?x rdfs:label "incomplete" .
?x rdfs:label "search" .

# q16
?x dblp:journal_name "SIGMOD" .
?x rdf:type dblp:Article .
?x rdfs:label "web" .
?x rdfs:label "search" .

# q17
?x rdf:type dblp:Article_in_Proceedings .
?x rdfs:label "semistructured" .
?x rdfs:label "search" .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "goldman" .

# q18
?x rdf:type dblp:Article_in_Proceedings .
?x rdfs:label "query" .
?x rdfs:label "cost" .
?x rdfs:label "optimization" .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "arvind" .

# q19
?x dblp:year "2007" .
?x rdfs:label "software" .
?x rdfs:label "time" .
?x rdf:type dblp:Article .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "zhu" .

# q20
?y rdf:type foaf:Person .
?y foaf:name "zhu" .
?y foaf:name "yuntao" .

# q21
?x dblp:year "2003" .
?x rdfs:label "data" .
?x rdfs:label "content" .
?x rdf:type dblp:Article_in_Proceedings .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "nikos" .

# q22
?x rdfs:label "spatial" .
?x rdf:type dblp:Article_in_Proceedings .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "jignesh" .

# q23
?x rdfs:label "algorithms" .
?x rdfs:label "parallel" .
?x rdfs:label "spatial" .
?x rdf:type dblp:Article_in_Proceedings .
?x dblp:author ?y .
?x dc:relation "conf" .
?y rdf:type foaf:Person .
?y foaf:name "patel" .

# q24
?x rdfs:label "implementation" .
?x rdfs:label "evaluation" .
?x rdf:type dblp:Article_in_Proceedings .
?x dblp:last_modified_date "2006-03-31" .
?x dblp:cites ?c .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "patel" .

# q25
?x rdfs:label "optimization" .
?x rdfs:label "query" .
?x rdf:type dblp:Article_in_Proceedings .
?x dblp:author ?y .
?x dblp:year "2003" .
?y rdf:type foaf:Person .
?y foaf:name ?n .

# q26
?x rdfs:label "xml" .
?x rdfs:label "tool" .
?x rdf:type dblp:Article_in_Proceedings .
?x dblp:year "2004" .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "patel" .

# q27
?x rdf:type dblp:Article_in_Proceedings .
?x rdfs:label "architecture" .
?x rdfs:label "web" .
?x dblp:last_modified_date "2005-09-05" .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "wu" .

# q28
?x rdf:type dblp:Article_in_Proceedings .
?x rdfs:label "language" .
?x rdfs:label "software" .
?x rdfs:label "system" .
?x dblp:year "2001" .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "roland" .

# q29
?x rdf:type dblp:Article_in_Proceedings .
```

```
?x rdfs:label "middleware" .
?x dblp:last_modified_date "2006-01-17" .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "sihvonen" .

# q30
?x rdf:type dblp:Article_in_Proceedings .
?x rdfs:label "middleware" .
?x rdfs:label "virtual" .
?x dblp:year "2001" .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "kwang" .

# q31
?x rdf:type dblp:Article .
?x rdfs:label "java" .
?x rdfs:label "code" .
?x rdfs:label "program" .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "roland" .

# q32
?x rdf:type dblp:Article .
?x rdfs:label "signal" .
?x rdfs:label "space" .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "zheng" .

# q33
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "fagin" .
?y foaf:name "roland" .

# q34
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "zheng" .
?y foaf:name "qui" .

# q35
?x rdf:type dblp:Article_in_Proceedings .
?x rdfs:label "processing" .
?x rdfs:label "query" .

# q36
?x rdf:type dblp:Article_in_Proceedings .
?x rdfs:label "xml" .
?x rdfs:label "processing" .

# q37
?x rdf:type dblp:Article_in_Proceedings .
?x rdfs:label "biological" .
?x rdfs:label "sequence" .
?x dblp:last_modified_date "2007-08-21" .
?x dblp:author ?y .
?y rdf:type foaf:Person .
?y foaf:name "jignesh" .

# q38
?x rdf:type dblp:Book .
?x rdfs:label "decision" .
?x rdfs:label "intelligent" .
?x rdfs:label "making" .
?x dc:publisher <http://www.springer.de/> .

# q39
?x rdf:type dblp:Proceedings .
?x rdfs:label "databases" .
?x rdfs:label "biological" .
?x dc:publisher <http://www.springer.de/> .

# q40
?x rdf:type dblp:Book .
?x rdfs:label "mining" .
?x rdfs:label "data" .

# q41
?x rdf:type dblp:Book .
?x rdfs:label "mining" .
?x rdfs:label "data" .
?x dc:publisher <http://www.springer.de/> .
?x dc:relation "trier.de" .
?x dc:relation "books" .

# q42
?x rdf:type dblp:Book .
?x rdfs:label "intelligence" .
?x rdfs:label "computational" .
?x dc:publisher <http://www.springer.de/> .
?x dc:relation "trier.de" .
?x dblp:year "2007" .

# q43
?x rdf:type dblp:Book .
?x rdfs:label "biologically" .
?x rdfs:label "inspired" .
?x rdfs:label "methods" .

# q44
?x rdf:type dblp:Book .
?x rdfs:label "networks" .
?x rdfs:label "neural" .

# q45
?x rdf:type dblp:Book .
?x rdfs:label "learning" .
?x rdfs:label "machine" .
?x dc:publisher <http://www.springer.de/> .

# q46
?x rdf:type dblp:Book .
?x rdfs:label "software" .
?x rdfs:label "system" .
?x dc:publisher <http://www.springer.de/> .

# q47
?x rdf:type dblp:Book .
?x rdfs:label "architecture" .
?x rdfs:label "computer" .

# q48
?x rdf:type dblp:Book .
?x rdfs:label "web" .
?x dblp:year "2006" .
?x dc:publisher ?p .
?x dblp:editor ?e .
?e foaf:name "kandel" .
?e foaf:name "abraham" .

# q49
?x rdf:type dblp:Book .
?x rdfs:label "theoretical" .
?x rdfs:label "science" .
?x dc:publisher <http://www.elsevier.nl/> .

# q50
?x rdf:type dblp:Book_Chapter .
```

```
?x rdfs:label "search" .
?x rdfs:label "semantic" .

# q51
?x rdf:type dblp:Article .
?x rdfs:label "search" .
?x rdfs:label "concept" .
?x rdfs:label "based" .

# q52
?x dblp:journal_name "sigmod" .
?x rdf:type dblp:Article .
?x rdfs:label "model" .
?x rdfs:label "information" .

# q53
?x dblp:journal_name "sigmod" .
?x rdf:type dblp:Article .
?x rdfs:label "dynamic" .
?x rdfs:label "networks" .

# q54
?x rdf:type dblp:Article_in_Proceedings .
?x rdfs:label "storage" .
?x rdfs:label "adaptive" .
?x dblp:author ?y .
?x dblp:year "2003" .
?y rdf:type foaf:Person .
?y foaf:name "jignesh" .
```

Listing 2. Queries for IMDB [23]

```
# @prefix imdb:
# <http://imdb/predicate/> .
# @prefix imdb_class:
# <http://imdb/class/> .
# @prefix rdf:
# <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

# q1
?x rdf:type imdb_class:name .
?x imdb:name "washington" .
?x imdb:name "denzel" .

# q2
?x rdf:type imdb_class:name .
?x imdb:name "eastwood" .
?x imdb:name "clint" .

# q3
?x rdf:type imdb_class:name .
?x imdb:name "john" .
?x imdb:name "wayne" .

# q4
?x rdf:type imdb_class:name .
?x imdb:name "smith" .
?x imdb:name "will" .

# q5
?x rdf:type imdb_class:name .
?x imdb:name "ford" .
?x imdb:name "harrison" .

# q6
?x rdf:type imdb_class:name .
?x imdb:name "julia" .
?x imdb:name "roberts" .

# q7
?x rdf:type imdb_class:name .
?x imdb:name "tom" .
?x imdb:name "hanks" .

# q8
?x rdf:type imdb_class:name .
?x imdb:name "johnny" .
?x imdb:name "depp" .

# q9
?x rdf:type imdb_class:name .
?x imdb:name "angelina" .
?x imdb:name "jolie" .

# q10
?x rdf:type imdb_class:name .
?x imdb:name "freeman" .
?x imdb:name "morgan" .

# q11
?x rdf:type imdb_class:title .
?x imdb:title "gone" .
?x imdb:title "with" .
?x imdb:title "the" .
?x imdb:title "wind" .

# q12
?x rdf:type imdb_class:title .
?x imdb:title "wars" .
?x imdb:title "star" .

# q13
?x rdf:type imdb_class:title .
?x imdb:title "casablanca" .

# q14
?x rdf:type imdb_class:title .
?x imdb:title "the" .
?x imdb:title "lord" .
?x imdb:title "rings" .

# q15
?x rdf:type imdb_class:title .
?x imdb:title "the" .
?x imdb:title "sound" .
?x imdb:title "music" .

# q16
?x rdf:type imdb_class:title .
?x imdb:title "wizard" .
?x imdb:title "oz" .

# q17
?x rdf:type imdb_class:title .
?x imdb:title "the" .
?x imdb:title "notebook" .

# q18
?x rdf:type imdb_class:title .
?x imdb:title "forrest" .
?x imdb:title "gump" .

# q19
?x rdf:type imdb_class:title .
?x imdb:title "the" .
?x imdb:title "princess" .
?x imdb:title "bride" .

# q20
?x rdf:type imdb_class:title .
?x imdb:title "the" .
```

```
?x imdb:title "godfather" .

# q21
?x imdb:title ?t .
?x rdf:type imdb_class:title .
?x imdb:cast_info ?z .
?r rdf:type imdb_class:char_name .
?r imdb:name "finch" .
?r imdb:name "atticus" .
?z rdf:type imdb_class:cast_info .
?z imdb:role ?r .

# q22
?x imdb:title ?t .
?x rdf:type imdb_class:title .
?x imdb:cast_info ?z .
?z rdf:type imdb_class:cast_info .
?r imdb:name "indiana" .
?r imdb:name "jones" .
?z imdb:role ?r .
?r rdf:type imdb_class:char_name .

# q23
?x imdb:title ?t .
?x rdf:type imdb_class:title .
?x imdb:cast_info ?z .
?z rdf:type imdb_class:cast_info .
?z imdb:role ?r .
?r rdf:type imdb_class:char_name .
?r imdb:name "james" .
?r imdb:name "bond" .

# q24
?x imdb:title ?t .
?x rdf:type imdb_class:title .
?x imdb:cast_info ?z .
?z rdf:type imdb_class:cast_info .
?z imdb:role ?r .
?r rdf:type imdb_class:char_name .
?r imdb:name "rick" .
?r imdb:name "blaine" .

# q25
?x imdb:title ?t .
?x imdb:cast_info ?z .
?z rdf:type imdb_class:cast_info .
?z imdb:role ?r .
?r rdf:type imdb_class:char_name .
?r imdb:name "kaine" .
?r imdb:name "will" .

# q26
?x imdb:title ?t .
?x rdf:type imdb_class:title .
?x imdb:cast_info ?z .
?z rdf:type imdb_class:cast_info .
?z imdb:role ?r .
?r rdf:type imdb_class:char_name .
?r imdb:name "dr." .
?r imdb:name "hannibal" .
?r imdb:name "lecter" .

# q27
?x imdb:title ?t .
?x rdf:type imdb_class:title .
?x imdb:cast_info ?z .
?z rdf:type imdb_class:cast_info .
?z imdb:role ?r .
?r rdf:type imdb_class:char_name .
?r imdb:name "norman" .
?r imdb:name "bates" .

# q28
?x imdb:title ?t .
?x rdf:type imdb_class:title .
?x imdb:cast_info ?z .
?z rdf:type imdb_class:cast_info .
?z imdb:role ?r .
?r rdf:type imdb_class:char_name .
?r imdb:name "darth" .
?r imdb:name "vader" .

# q29
?x imdb:title ?t .
?x rdf:type imdb_class:title .
?x imdb:cast_info ?z .
?z rdf:type imdb_class:cast_info .
?z imdb:role ?r .
?r rdf:type imdb_class:char_name .
?r imdb:name "the␣" .
?r imdb:name "wicked" .
?r imdb:name "witch" .
?r imdb:name "the" .
?r imdb:name "west" .

# q30
?x imdb:title ?t .
?x rdf:type imdb_class:title .
?x imdb:cast_info ?z .
?z rdf:type imdb_class:cast_info .
?z imdb:role ?r .
?r rdf:type imdb_class:char_name .
?r imdb:name "nurse" .
?r imdb:name "ratched" .

# q31
?x imdb:title ?t .
?x rdf:type imdb_class:title .
?x imdb:movie_info ?i .
?i rdf:type imdb_class:movie_info .
?i imdb:info "frankly" .
?i imdb:info "dear" .
?i imdb:info "don't" .
?i imdb:info "give" .
?i imdb:info "damn" .

# q32
?x imdb:title ?t .
?x rdf:type imdb_class:title .
?x imdb:movie_info ?i .
?i rdf:type imdb_class:movie_info .
?i imdb:info "going" .
?i imdb:info "make" .
?i imdb:info "offer" .
?i imdb:info "can't" .
?i imdb:info "refuse" .

# q33
?x imdb:title ?t .
?x rdf:type imdb_class:title .
?x imdb:movie_info ?i .
?i rdf:type imdb_class:movie_info .
?i imdb:info "understand" .
?i imdb:info "class" .
?i imdb:info "contender" .
?i imdb:info "coulda" .
?i imdb:info "somebody" .
?i imdb:info "instead" .
?i imdb:info "bum" .

# q34
?x imdb:title ?t .
```

```
?x rdf:type imdb_class:title .
?x imdb:movie_info ?i .
?i rdf:type imdb_class:movie_info .
?i imdb:info "toto" .
?i imdb:info "feeling" .
?i imdb:info "not" .
?i imdb:info "kansas" .
?i imdb:info "anymore" .

# q35
?x imdb:title ?t .
?x rdf:type imdb_class:title .
?x imdb:movie_info ?i .
?i rdf:type imdb_class:movie_info .
?i imdb:info "here's" .
?i imdb:info "looking" .
?i imdb:info "kid" .

# q36
?x rdf:type imdb_class:title .
?c rdf:type imdb_class:cast_info .
?x imdb:cast_info ?c .
?c imdb:role ?r .
?r rdf:type imdb_class:char_name .
?r imdb:name "skywalker" .
?c imdb:person ?p .
?p rdf:type imdb_class:name .
?p imdb:name "hamill" .

# q37
?x imdb:year "2004" .
?x rdf:type imdb_class:title .
?x imdb:title ?t .
?x imdb:cast_info ?c .
?c rdf:type imdb_class:cast_info .
?c imdb:person ?p .
?p rdf:type imdb_class:name .
?p imdb:name "hanks" .

# q38 #
?r imdb:name ?rn .
?r rdf:type imdb_class:char_name .
?x rdf:type imdb_class:title .
?x imdb:title "yours" .
?x imdb:title "mine" .
?x imdb:title "ours" .
?x imdb:cast_info ?c .
?c rdf:type imdb_class:cast_info .
?c imdb:role ?r .
?c imdb:person ?p .
?p rdf:type imdb_class:name .
?p imdb:name "henry" .
?p imdb:name "fonda" .

# q39
?x rdf:type imdb_class:title .
?x imdb:title "gladiator" .
?x imdb:cast_info ?c .
?c rdf:type imdb_class:cast_info .
?c imdb:role ?r .
?r imdb:name ?rn .
?r rdf:type imdb_class:char_name .
?c imdb:person ?p .
?p rdf:type imdb_class:name .
?p imdb:name "russell" .
?p imdb:name "crowe" .

# q40
?x rdf:type imdb_class:title .
?x imdb:title "star" .
?x imdb:title "trek" .
?x imdb:cast_info ?c .
?r rdf:type imdb_class:char_name .
?r imdb:name ?rn .
?c rdf:type imdb_class:cast_info .
?c imdb:role ?r .
?c imdb:person ?p .
?p rdf:type imdb_class:name .
?p imdb:name "spiner" .
?p imdb:name "brent" .

# q41
?x imdb:year "1951" .
?x imdb:title ?t .
?x rdf:type imdb_class:title .
?x imdb:cast_info ?c .
?c rdf:type imdb_class:cast_info .
?c imdb:person ?p .
?p rdf:type imdb_class:name .
?p imdb:name "audrey" .
?p imdb:name "hepburn" .

# q42
?p rdf:type imdb_class:name .
?p imdb:name ?n .
?c imdb:person ?p .
?c rdf:type imdb_class:cast_info .
?c imdb:role ?r .
?r rdf:type imdb_class:char_name .
?r imdb:name "jacques" .
?r imdb:name "clouseau" .

# q43
?p rdf:type imdb_class:name .
?p imdb:name ?n .
?c imdb:person ?p .
?c rdf:type imdb_class:cast_info .
?c imdb:role ?r .
?r rdf:type imdb_class:char_name .
?r imdb:name "jack" .
?r imdb:name "ryan" .

# q44
?p rdf:type imdb_class:name .
?p imdb:name "stallone" .
?c imdb:person ?p .
?c rdf:type imdb_class:cast_info .
?c imdb:role ?r .
?r rdf:type imdb_class:char_name .
?r imdb:name "rocky" .

# q45
?p rdf:type imdb_class:name .
?p imdb:name ?n .
?c imdb:person ?p .
?c rdf:type imdb_class:cast_info .
?c imdb:role ?r .
?r rdf:type imdb_class:char_name .
?r imdb:name "terminator" .

# omitted q46 to q49

# q50
?a rdf:type imdb_class:title .
?a imdb:title "lost" .
?a imdb:title "ark" .
?a imdb:cast_info ?ca .
?ca rdf:type imdb_class:cast_info .
?ca imdb:person ?p .
?p rdf:type imdb_class:name .
?p imdb:name ?n .
?ci rdf:type imdb_class:cast_info .
```

```
?ci imdb:person ?p .
?i rdf:type imdb_class:title .
?i imdb:cast_info ?ci .
?i imdb:title "indiana" .
?i imdb:title "jones" .
?i imdb:title "last" .
?i imdb:title "crusade" .
```