Topic-based Selectivity Estimation for Hybrid Queries over RDF Graphs
Technical Report

Andreas Wagner[†], Veli Bicer[‡], and Thanh Tran[†]

[†] Karlsruhe Institute of Technology and [‡] IBM Research Centre Dublin
{a.wagner,thanh.tran}@kit.edu, velibice@ie.ibm.com

Abstract. Many RDF descriptions today are *text-rich*: besides *structured* data they also feature much *unstructured* text. Text-rich RDF data is frequently queried via predicates matching structured data, combined with string predicates for textual constraints (*hybrid queries*). Evaluating hybrid queries efficiently requires means for *selectivity estimation*. Previous works on selectivity estimation, however, suffer from inherent drawbacks, which are reflected in efficiency and effectiveness issues. We propose a novel estimation approach, *TopGuess*, which exploits topic models as data synopsis. This way, we capture correlations between structured and unstructured data in a *uniform and scalable* manner. We study TopGuess in a theoretical analysis and show it to guarantee a linear space complexity w.r.t. text data size. Further, we show selectivity estimation time complexity to be independent from the synopsis size. In experiments on real-world data, TopGuess allowed for great improvements in estimation accuracy, without sacrificing efficiency.

1 Introduction

RDF features descriptions of entities, with each description being a set of *triples*. A triple $\langle s, p, o\rangle$ associates an entity (subject) s with an *object* o via a *predicate* p. Many RDF descriptions feature *text data*. Reasons are twofold: On the one hand, structured RDF often comprises text via predicates such as `comment` or `description`. On the other hand, unstructured Web documents are frequently annotated with structured data (e.g., via RDFa or Microformats).[1]

Hybrid Queries. Such text-rich RDF descriptions are often queried with *hybrid queries* – comprising predicates that match *structured data* as well as *words* in text data. Consider the following example, cf. Fig. 1-a/b:

```
SELECT * WHERE {
   ?m ex:title ?title .    ?p ex:name ?name .    ?l ex:name ?name2 .
   ?m ex:starring ?p .     ?p ex:bornIn ?l .
   ?m rdf:type Movie .     ?p rdf:type Person .
   FILTER (contains(?title,"Holiday") && contains(?name,"Audrey") &&
           contains(?name2,"Belgium")) }
```

[1] `http://www.webdatacommons.org`

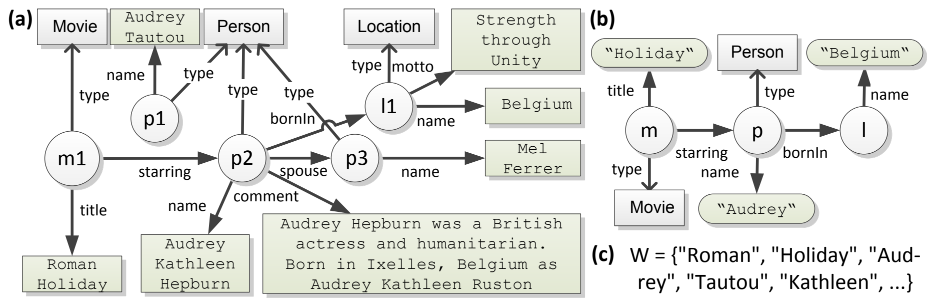


Fig. 1. (a) RDF graph about "Audrey Hepburn" and her movie "Roman Holiday". (b) Hybrid query graph asking for movies with title "Holiday" and starring "Audrey". (c) Vocabulary W comprising all words from attributes values in the data graph.

Hybrid queries are highly relevant for RDF stores with SPARQL fulltext extension, e.g., LARQ[2] or Virtuoso[3], or databases with text search, e.g., [16]. *In fact, every store that supports* `FILTER` *clauses on texts faces hybrid queries.*

Selectivity Estimation. For finding an optimal query plan, RDF stores rely on *selectivity estimates* to approximate the result size for a query (fragment). Selectivity estimation techniques use *data synopses* [9,20,21] to effectively capture data correlations and efficiently compute query selectivities [7,10,17,19].

Unfortunately, with regard to hybrid queries, state-of-art selectivity estimation approaches suffer from two major shortcomings: (I.1) *Effectiveness Issues.* Previous works utilize data synopses that either do not capture text data, e.g., [9,20], or exploit separate synopses for text and structural data [21]. However, both cases may lead to inaccurate selectivity estimates, because correlations between structure and text data are missed. (I.2) *Efficiency Issues.* While our previous work [21] captures correlations in text and structured data, it suffers from efficiency problems given long texts/many words in the data. That is, the synopsis size as well as the selectivity estimation performance is directly coupled with the number of words included in the synopsis. Heuristics must be employed to scale this approach to large a vocabulary of words.

In contrast, we propose *TopGuess*, which utilizes *relational topic models* as data synopsis. Such synopses not only incorporate text, but also capture correlations with structured data [2,4,23]. This way, we have *one single synopsis for text and structured data*. Further, TopGuess omits above efficiency issues via a *small and query-specific* Bayesian network. In fact, our estimation time complexity is *independent of the synopsis size.*

Contributions. (1) We propose TopGuess, which features relational topic models as synopsis. Our approach allows for a holistic learning of correlations in text as well as structured data. To the best of our knowledge, *this is the first selectivity estimation work to use topic models as data synopsis.* (2) We provide a theoretical analysis: We show TopGuess to achieve a *linear space complexity* w.r.t. text data size, cf. Thm. 1. Further, we show its estimation time

[2] `http://jena.sourceforge.net/ARQ/lucene-arq.html`
[3] `http://virtuoso.openlinksw.com`

complexity to be *independent of the synopsis size*, cf. Thm. 4. (3) We conducted experiments on real-world data. Our results suggest that estimation effectiveness can be improved by up to 88%, without sacrificing runtime performance.

Outline. First, in Sect. 2 we outline preliminaries. We introduce the TopGuess approach in Sect. 3. In Sect. 4, we present evaluation results, before we outline related work in Sect. 5. We conclude with Sect. 6.

2 Preliminaries

Data and Query Model. We use RDF as data model:

Definition 1 (RDF Graph). *Let ℓ_a (ℓ_r) be a set of attribute (relation) labels. A RDF graph is a directed labeled graph $\mathcal{G} = (V, E, \ell_a, \ell_r)$, with nodes $V = V_E \uplus V_A \uplus V_C$ where V_E are entity nodes, V_A are attribute value nodes, and V_C are class nodes. Edges (so-called triples) $E = E_R \uplus E_A \uplus type$ are a disjoint union of relation edges E_R and attribute edges E_A. Relation edges connect entity nodes: $\langle s, r, o\rangle \in E_R$, with $s, o \in V_E$ and $r \in \ell_r$). Attribute edges connect an entity with an attribute value: $\langle s, a, o\rangle \in E_A$, with $s \in V_E, o \in V_A$ and $a \in \ell_a$. Triple $\langle s, type, o\rangle \in E$ models that entity $s \in V_E$ belongs to class $o \in V_C$.*

We conceive an attribute value in V_A as a *bag-of-words*. Further, let a vocabulary W comprise all such words. That is, W is 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. See also Fig. 1.

Conjunctive queries resemble the basic graph pattern (BGP) feature of SPARQL. In this work, we use *hybrid queries*:

Definition 2 (Hybrid Query). *A query Q is a directed graph $G_Q = (V_Q, E_Q)$, with $V_Q = V_{Q_V} \uplus V_{Q_C} \uplus V_{Q_K}$, V_{Q_V} as variables, V_{Q_C} as constants, and V_{Q_K} as keywords. Edges E_Q are called predicates: (1) Class predicates $\langle s, type, o\rangle$, with $s \in V_{Q_V}, o \in V_{Q_C}$. (2) Relation predicates $\langle s, r, o\rangle$, with $s \in V_{Q_V}$, $o \in V_{Q_C} \uplus V_{Q_V}$, and $r \in \ell_r$. (3) String predicates $\langle s, a, o\rangle$, with $s \in V_{Q_V}$, $o \in V_{Q_K}$, and $a \in \ell_a$.*

Fig. 1-b shows an example query. Query semantics follow those for BGPs. That is, results are subgraphs of the data graph, which match all query predicates. 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.

We rely on two data synopses: *topic models* and *Bayesian networks* (BNs):

Topic Models. Topic models assume that texts are mixtures of "hidden" topics, where a topic is a probability distribution over words. These topics are abstract clusters of words – formed according to word co-occurrences. More formally, a text collection can be represented by k topics $\mathcal{T} = \{t_1, \ldots, t_k\}$, where W is the vocabulary (see above definition) and each topic $t \in \mathcal{T}$ is a multinomial distribution of words in W: $\mathcal{P}(w \mid t) = \beta_{tw}$ and $\sum_{w \in W} \beta_{tw} = 1$.

Example. Three topics are depicted in Fig. 2-c: $\mathcal{T} = \{t_1, t_2, t_3\}$. Every topic t assigns a probability (represented by vector β_t) to each word in the vocabulary. Probabilities in β_t indicate the importance of words within topic t. For instance, "Belgium" is most important for topic t_3 ($\beta_{tw} = 0.014$).

Bayesian Networks. A Bayesian network (BN) is a directed graphical model, which compactly represents a joint probability distribution via its structure and parameters [12]. The structure is a directed acyclic graph, where nodes stand for random variables and edges represent dependencies. Given a node X_i and its parents $\mathbf{Pa}(X_i) = \{X_j, \ldots, X_k\}$, X_i dependents on $\mathbf{Pa}(X_i)$, but is *conditionally independent* of all non-descendant random variables (given $\mathbf{Pa}(X_i)$).

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 $\mathcal{P}(X_i \mid \mathbf{Pa}(X_i))$. The joint distribution $\mathcal{P}(X_1, \ldots, X_n)$ can be estimated via the chain rule [12]: $\mathcal{P}(X_1, \ldots, X_n) \approx \prod_i \mathcal{P}(X_i \mid \mathbf{Pa}(X_i))$.

Example. A BN is shown in Fig. 3. Nodes such as X_m and $X_{holiday}$ stand for random variables. Edges stand for dependencies between those nodes. For instance, the edge $X_m \rightarrow X_{holiday}$ denotes a dependency between the parent, X_m, and the child, $X_{holiday}$. In fact, given its parent, $X_{holiday}$ is conditionally independent of all non-descendant variables, e.g., X_p. Every node has a CDP. For example, $X_{holiday}$ has a CDP for $\mathcal{P}(X_{holiday} \mid X_m)$, cf. Fig. 3-b.

Problem. Given a hybrid query Q, we aim at an result size estimation function $\mathcal{F}(Q)$ as [9]: $\mathcal{F}(Q) \approx \mathcal{R}(Q) \cdot \mathcal{P}(Q)$.

Let $\mathcal{R}$ be a function $\mathcal{R} : Q \rightarrow \mathbb{N}$ that provides an *upper bound cardinality* for a result set for query Q. Further, let $\mathcal{P}$ be a *probabilistic component*, which assigns a probability to query Q that models whether Q's result set is non-empty.

$\mathcal{R}(Q)$ can be easily computed as product over "class cardinalities" of Q [9]. 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. Finally, $\mathcal{R}(Q) = \prod_v R(v)$.

In the remainder of the paper, we provide an effective (I.1) and efficient (I.2) instantiation of the probabilistic component $\mathcal{P}$.

3 TopGuess

Targeting the effectiveness (I.1) and efficiency (I.2) issues of existing works w.r.t. hybrid queries (cf. Sect. 1), we now introduce our novel TopGuess approach.

More precisely, we present a *uniform data synopsis based on relational topic models* in Sect. 3.1 (I.1), and show in Thm. 1 that this synopsis has a *linear space complexity* w.r.t. vocabulary W (I.2). Further, we introduce a probabilistic component $\mathcal{P}$ in Sect. 3.2+3.3, and show in Thm. 4 selectivity computation complexity to be *independent of the synopsis size* (I.2).

Note, the topic model (data synopsis) is learned at *offline time* and may be stored on disk. At *runtime*, we construct a small BN for each given query – *reflecting our data synopsis as well as query characteristics via topic mixtures.*

3.1 Relational Topic Models as Data Synopsis

Synopsis Parameters. For an effective synopsis over text-rich RDF data, TopGuess exploits *relational topic models* [2,4,14,23]. These topic models summarize the data by means of one *uniform synopsis* – considering structured and text data. More precisely, our synopsis comprises of two parts:

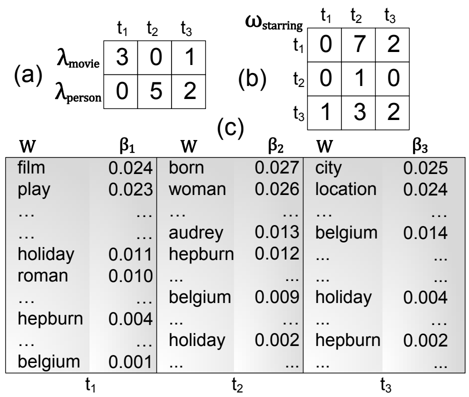


Fig. 2. Synopsis parameters (stored on disk): (a) λ_{movie} and λ_{person} parameter for three topics. (b) ω matrix for `starring` relation and three topics – rows (columns) represent source (target) topics of the relation. (c) Words from the vocabulary W and their corresponding probabilities for each topic, β_t. Note, data is taken from Fig. 1-a/c.

(1) First, the TopGuess synopsis captures text data in a *low-dimensional representation* via a set of k topics $\mathcal{T} = \{t_1, \ldots, t_k\}$.

Example. Fig. 2-c groups words from the vocabulary $W = \{$ "Roman", "Holiday", $\ldots\}$, cf. Fig. 1-c, via three topics, $\mathcal{T} = \{t_1, t_2, t_3\}$. This way, text data in Fig. 1-a, e.g., associated via attribute `comment`*, is compactly represented.*

Depending on data characteristics, e.g., the amount of text data, a small number of topics (e.g., 50) may already capture meaningful correlations in the data. By means of this compact summary, TopGuess achieves a linear space complexity linear w.r.t. a vocabulary, see Thm. 1. Notice, previous works allow to learn the optimal number of topics [8].

(2) Second, the TopGuess synopsis captures *correlations between those topics and structured data*. For our query model, we rely on two correlation parameters for selectivity estimation: λ and ω. Note, for other kinds of queries, further types of correlation parameters may be considered.

– *Class-Topic Parameter* λ. We capture correlations between a class $c \in V_C$ and topics in $\mathcal{T}$ via a vector λ_c, where each vector element, $\lambda_c(t)$, represents the weight between class c and topic t. A higher weight indicates that a class is more correlated with a topic.
  Example. Fig. 2-a shows the two class-topic parameters for the classes `Movie` *and* `Person`*: λ_{movie} and λ_{person}. Both indicate correlations w.r.t. topics $\mathcal{T} = \{t_1, t_2, t_3\}$. For instance, λ_{movie} states that class* `Movie` *is highly correlated with topic t_1, has some correlation with t_3, and has no correlation with t_2.*
– *Relation-Topic Parameter* ω. We measure correlations between a relation r and the topics in $\mathcal{T}$ via a matrix ω_r. Since relation r is observed "between" two entities, say $\langle s, r, o\rangle$, the topic of its subject s and its object o is considered. Given k topics, matrix ω_r has $k \times k$ dimensions and entries such that: if entity s is associated with topic t_i and entity o has topic t_j, the weight of observing a relation r "between" s and o is given by the entry $\langle i, j\rangle$, denoted as $\omega_r(t_i, t_j)$. Note, TopGuess features a matrix ω for each relation.
  Example. Fig. 2-b depicts the relation-topic parameter for the `starring` *relation: $\omega_{starring}$. According to $\omega_{starring}$,* `starring` *is most often observed (weight 7) if its subject (object) contains words from topic t_1 (t_2).*

Parameter Learning. For training above parameters, *we do not restrict TopGuess to a single topic model. Instead, different approaches can be used.* For instance, classical topic models such as LDA [3] may be employed to learn the first part, i.e., word/topic probabilities, cf. Fig. 2-c. Then, correlations between those topics and classes/relations must be obtained. For this, topic models have been extended to consider structured data, so-called relational topic models [2,4,14,23]. Most notably, a recent approach, the *Topical Relational Model* (TRM) [2], trains topics as well as class/relation correlations simultaneously from RDF data. We used a TRM as data synopsis in our experiments.

Discussion. The TopGuess synopsis comes with key advantages: First, in contrast to existing work [21], we do not need separate synopses for structured and text data. This way, we may learn correlations in a uniform manner.

Moreover, TopGuess parameters are not required to be loaded in memory. This is a crucial advantage over state-of-the-art selectivity estimation systems [9,20,21], as memory is commonly a limited resource. So, TopGuess can utilize the *complete vocabulary* W for learning word/topic probabilities β.

Last, as empirically validated by our experiments, correlations between topics and structured data suffice for an accurate selectivity estimation. Since even a small number of topics can capture these correlations, our synopsis does not grow exponentially in its vocabulary size.

In fact, we can show that a topic-based data synopsis has linear space complexity w.r.t. its vocabulary:

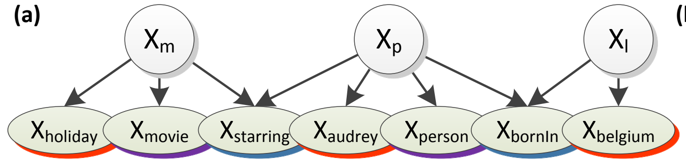
Theorem 1 (Synopsis Space Complexity). *Given k topics, a vocabulary W, classes V_C, and relations ℓ_r, TopGuess has a storage space complexity in $O(|W| \cdot k + |V_C| \cdot k + |\ell_r| \cdot k^2)$.*

Proof. For each topic, we store probabilities of every word in W. So, the complexity of K topics is $O(|W| \cdot K)$. $\boldsymbol{\lambda}$ is a matrix with $|V_C| \times K$ dimensions, which associates classes with topics. Thus, its complexity is in $O(|V_C| \cdot K)$. Every relation in ℓ_r is represented as $K \times K$ matrix, which has a space complexity in $O(K^2)$. Overall, our synopsis space complexity is in $O(|W| \cdot K + |V_C| \cdot K + |\ell_r| \cdot K^2)$ ■

3.2 Probabilistic Component: Query-Specific BN

In this section, we exploit the synopsis parameters for an efficient probabilistic component (I.2, Sect. 1). For this, we first construct a small, query-specific BN and afterwards compute its joint probability in Sect. 3.3. Both steps are done at runtime. *In contrast to [9,20,21], all synopsis parameters may be kept on disk, while only the query-specific BN is loaded into memory.*

To construct a BN specifically for a query Q, we capture every query predicate in Q via a random variable: For a class predicate, $\langle s, type, c\rangle$, and relation predicate, $\langle s, r, o\rangle$, we create a binary random variable X_c and X_r. Similarly, for a string predicate, $\langle s, a, w\rangle$, we introduce a binary random variable X_w. Most importantly, we assume that each variable v in Q belongs to one or more topics in $\mathcal{T}$. So, we model variable v via a *topical random variable*, X_v. More formally, X_v has a multinomial distribution over the topics:



(b) CPD: $P(X_{holiday} | X_m)$

$X_{holiday}$ = **T**	X_m
0.647	t_1
0.118	t_2
0.235	t_3

Fig. 3. (a) Query-specific BN for the query in Fig.1-b. It contains three topical variables (color: gray, e.g., X_m), two class predicate variables (color: pink, e.g., X_{movie}), two relation predicate variables (color: blue, e.g., $X_{starring}$), and three string predicate variables (color: red, e.g., $X_{holiday}$). Observed variables (pink/blue/red) are independent from each other and only dependent on hidden topical random variables (gray) – as dictated by Def. 4. (b) CDP for random variable $X_{holiday}$, cf. parameters in Fig. 2-c.

Definition 3 (Topical Random Variable). *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. Thus, X_v captures query variable v's "relatedness" to every topic. In the following, we denote the set of all string, class, relation, and topical random variables for query Q as $\mathbf{X}_w$, $\mathbf{X}_c$, $\mathbf{X}_r$, and $\mathbf{X}_v$.

We create a simple BN structure by means of a fixed structure assumption:

Definition 4 (Topical Dependence Asmp.). *Given a class/string predicate $\langle v, *, * \rangle$, the corresponding variable X depends only the topical random variable X_v. Given a relation predicate $\langle v, *, y \rangle$, the corresponding variable X depends only on two topical random variables: X_v and X_y.*

The topical dependence assumption lies at the core of the TopGuess approach. *It considers that query predicate probabilities depend on (and are governed by) the topics of their associated topical random variables.* Further, the assumption allows us to model the query probability, $\mathcal{P}(Q)$, via a tree-shaped BN.

Example. Fig. 3-a depicts a BN for the query in Fig. 1-b. Adhering to Def. 4, each topical variable (X_m, X_p, and X_l) forms a small tree of dependent random variables. For instance, random variable $X_{holiday}$ is only dependent on its topical variable, X_m. In fact, given X_m, $X_{holiday}$ is conditionally independent of all other variables, e.g., X_{audrey}. This way, topic mixtures of X_m, X_p, and X_l govern the overall query probability, $\mathcal{P}(Q)$.

Last, note that the topical dependence assumption leads to a *valid* BN:

Theorem 2. *A query-specific BN constructed according to Def. 4 is acyclic.*

Proof. 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. 3-a). Thus, no cycle can be introduced ■

3.3 Probabilistic Component: Query Probability Computation

Having formed the BN structure for a given query Q, we may compute the query probability, $\mathcal{P}(Q)$, via the chain rule (CR) [12]:

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

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

In order to solve Eq. 1 we a require CPD for each random variable, cf. Fig. 3-b. We rely on TopGuess parameters as well as topic distributions of topical random variables to approximate these CPDs. As topical variables $\mathbf{X}_v$ are hidden, we learn their distributions from observed random variables ($\mathbf{X}_w$, $\mathbf{X}_c$, $\mathbf{X}_r$).

In the following, we first discuss CPD estimation for observed random variables, given topical random variables (topic distributions). Subsequently, we present learning of topic distributions for hidden topical variables.

Query Predicate Probabilities. Probabilities for query predicates are influenced by their associated topical random variables and their TopGuess parameters. In other words, we may compute the CPDs for $\mathbf{X}_w$, $\mathbf{X}_c$, and $\mathbf{X}_r$ using topic distributions of topical variables and probabilities estimated by the corresponding β, λ, or ω parameter:

(1) *Class Predicates.* Adhering to the topical dependence assumption, the probability of observing a class, $\mathcal{P}(X_c = \mathbf{T})$, is only dependent on its topical variable X_v. We use the class-topic parameter λ to obtain the weight $\lambda_c(t)$, which indicates the correlation between topic t and class c:

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

Example. Fig. 3-a shows two random variables, X_{movie} and X_{person}, which dependent on their topical variables X_m and X_p. For computing $\mathcal{P}(X_{movie} = \mathbf{T})$ and $\mathcal{P}(X_{person} = \mathbf{T})$, the parameters λ_{movie} and λ_{person} are used, cf. Fig. 2-a. Assuming $\mathcal{P}(X_m = t_1) = 0.6$, $\mathcal{P}(X_m = t_2) = 0.1$, and $\mathcal{P}(X_m = t_3) = 0.3$, we get: $\mathcal{P}(X_{movie} = \mathbf{T}) = 0.6 \cdot 0.75 + 0.1 \cdot 0 + 0.3 \cdot 0.25 = 0.525$.

(2) *Relation Predicates.* A relation predicate $\langle v, r, y\rangle$ connects two query variables, which have the two topical variables X_v and X_y. Random variable X_r solely dependents on the topics of X_v and X_y. The correlation between relation r and these topics is given by the relation-topic parameter ω_r:

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

Example. In Fig. 3-a, we have the variables $X_{starring}$ and X_{bornIn} – both dependent on two topical variables. For instance, $X_{starring}$ depends on X_m and X_p. $\mathcal{P}(X_{starring} = \mathbf{T})$ is estimated via matrix $\omega_{starring}$, cf. Fig. 2-b.

(3) *String Predicates.* For each string predicate $\langle v, a, w\rangle$, there is a random variable X_w. The word-topic parameter β_{tw} represents the probability of observing word w given topic t. Thus, $\mathcal{P}(X_w = \mathbf{T})$ is calculated as probability of observing w, given the topics of v's topical variable, X_v:

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

Example. Fig. 3-a depicts three random variables for string predicates. Given $\mathcal{P}(X_m)$ as in the above example, the probability for "holiday" is (cf. Fig. 3-b):

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

Learning Topic Distributions. Finally, we wish to estimate topic distributions for the hidden topical variables based on $\mathbf{X}_w$, $\mathbf{X}_c$, and $\mathbf{X}_r$. *We aim at finding a topic distribution for every topical variable, so that the query probability in Eq. 1 is maximized. Thus, this optimal topic distribution directly gives us* $\mathcal{P}(Q)$. Let θ_{vt} denote a set of topic parameters for topical random variable X_v. Further, let $\theta = \{\theta_{vt} \mid v \in V_{Q_V}, t \in \mathcal{T}\}$ be the set of all parameters θ_{vt}. Then, we search for parameter θ that maximizes the log-likelihood of Eq. 1:

$$\arg\max_{\theta} \ \ell(\theta : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r) \tag{2}$$

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

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

where $\mathbf{X}_w^v$ and $\mathbf{X}_c^v$ is the set of all string/class random variables having X_v as parent. $\mathbf{X}_r^{v,y}$ is the set of all relation random variables with parents X_v and X_y.

We use gradient ascent optimization to learn the parameter θ. First, we parametrize each $\mathcal{P}(X_v = t)$ with θ_{vt} such that

$$\mathcal{P}(X_v = t) = \frac{e^{\theta_{vt}}}{\sum_{t' \in \mathcal{T}} e^{\theta_{vt'}}}$$

to obtain a valid 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 [12]:

Theorem 3. *Given a BN and $\mathcal{D} = \{\boldsymbol{o}[1], \ldots, \boldsymbol{o}[M]\}$ as a partially observed dataset. Let X be a variable in that BN with $\mathbf{Pa}(X)$ as its parents. Then:*

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

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

$$\frac{\partial \ell(\theta : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r)}{\partial \theta_{vt}} = \underbrace{\frac{\partial \ell(\theta : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r)}{\partial \mathcal{P}(X_v = t)}}_{\text{(i)}} \cdot \underbrace{\frac{\partial \mathcal{P}(X_v = t)}{\partial \theta_{vt}}}_{\text{(ii)}} \tag{3}$$

The (i)-part of the gradient, Eq. 3, may be obtained via Theorem 3:

$$\frac{\partial \ell(\theta : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r)}{\partial \mathcal{P}(X_v = t)} = \frac{1}{\mathcal{P}(X_v = t)} \Bigg(\sum_{X_w \in \mathbf{X}_w^v} \mathcal{P}(X_v = t \mid X_w, \beta)$$

$$+ \sum_{X_c \in \mathbf{X}_c^v} \mathcal{P}(X_v = t \mid X_c, \lambda) + \sum_{y} \sum_{X_r \in \mathbf{X}_r^{v,y}} \mathcal{P}(X_v = t \mid X_r, X_y, \omega) \Bigg)$$

Using the Bayes rule we get:

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

Finally, the (ii)-part of the gradient in Eq. 3 is given by:

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

Time Complexity. Query probability estimation has a complexity bound:

Theorem 4 (Time Complexity for $\mathcal{P}(Q)$ Estimation). *Given k topics and a query Q, the time for computing $\mathcal{P}(Q)$ in Eq. 1 is in $O(\psi \cdot |Q| \cdot k)$, with ψ as number of iterations needed for optimization and $|Q|$ as # predicates in Q.*

Proof. Complexity for $\mathcal{P}(Q)$ is comprised of: (1) Estimation time for the joint probability of Q's query-specific BN. (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. So, overall complexity for computing $\mathcal{P}(Q)$ is *independent of the synopsis size.*

4 Evaluation

We conducted experiments to analyze the effectiveness (I.1) and efficiency (I.2) of TopGuess. Overall, our results are very promising: we achieved up to 88% more accurate selectivity estimates, while runtime was comparable to the baselines. Further, in contrast to the baselines, we noted TopGuess's runtime behavior to be much less influenced by the synopsis size – thereby confirming Thm. 4.

4.1 Evaluation Setting

Systems. We employ two categories of baselines: (1) String predicates are combined with structured predicates via an *independence assumption*: `ind` baseline. That is, the selectivity of string predicates and structured predicates is estimated using two separate synopses: a string synopsis (explained below) and a histogram [10]. Obtained probabilities are combined in a greedy fashion while assuming independence. (2) We reuse our previous work on BNs for text-rich data graphs [21]: `bn` baseline. Here, *all query predicates are captured uniformly via a single BN*. To handle sting predicates, we employ *n-gram string synopses* [22]

A n-gram synopsis summarizes the vocabulary by "omitting" certain n-grams. Thus, a synopsis represents a subset of all possible n-grams occurring in the data. 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* (sbf) synopsis has been proposed [22], which uses bloom filters as a heuristic map that projects n-grams to their counts. Note, we refer to omitted 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. So, a string predicate featuring a missing n-gram is assumed to be independent from the remainder of the query. Its probability is computed via a heuristic. We employ the *leftbackoff* strategy, which finds the longest known n-gram that is the pre- or postfix of the missing n-gram. Then, the probability of the missing n-gram is approximated based on statistics for its pre-/postfix [22].

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

Data. We employ two real-world RDF datasets: DBLP [15] and IMDB [6]. From both datasets we have large vocabularies: $25,540,172$ (DBLP) and $7,841,347$ (IMDB) words. Note, while DBLP and IMDB feature text-rich attributes, they differ in their overall amount of text. On average an attribute in DBLP contains 2.6 words, with a variance of 2.1 words. In contrast, IMDB attributes contain 5.1 words, with a variance of 95.6 words. Moreover, we observed during learning of the `bn` baseline that there are more data correlations in IMDB than in DBLP. *We expect correlations between query predicates to have a strong influence on the system effectiveness.*

Queries. We used IMDB [6] and DBLP [15] keyword search benchmarks: We generated 54 DBLP queries from [15]. Further, we constructed 46 queries

	IMDB		DBLP	
	`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

Table 1. Data synopsis memory/disk space in MB.

for IMDB based on queries in [6]. We omitted 4 queries from [6], as they could not be translated to conjunctive queries. Overall, our load features 100 queries with: 0-4 relation, 1-7 string, 1-4 class predicates, and 2-11 predicates in total. Further query statistics as well as a complete query listing is given in Sect. 7.

Synopsis Size. We employ baselines with varying synopsis size. For this, we varied # words captured by the string synopsis. The top-k and sample synopsis contained # words $\in \{0.5K, 1K, 5K, 10K\}$. The sbf string synopsis captured $\{2.5K, 5K, 25K, 50K\}$ words for each attribute. Note, sbf systems featured most keywords occurring in our query load. Different string synopsis sizes translated to a memory consumption of baselines $\in$ {2, 4, 20, 40} MB. `ind` and `bn` baselines load their synopsis into main memory. In contrast, TopGuess keeps a large topic model at disk and constructs a small, query-specific BN in memory at runtime ($\leq$ 100 KBytes). Table 1 depicts further details.

Implementation and Offline Learning. For `ind` and `bn` baselines, we started by constructing their string synopses. Each synopsis was learned in $\leq$ 1h.

Then, we constructed `bn` systems based on [21]. That is, we capture words and structured data elements using random variables and learn correlations between them, thereby forming a BN 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 `ind` systems, we do not need the model structure and merely keep the marginalized `bn` parameters. Structure and parameter learning combined took up to 3h. To compute query selectivities the `bn` systems need inferencing strategies. For this, we used a Junction tree algorithm [21].

TopGuess exploits an "off-the-shelf" TRM from [2]. The number of topics is an important factor – determining which correlations are discovered. We experimented with a varying number of topics $\in [10, 100]$. We found 50 topics are sufficient to capture all strong correlations in our datasets. The TopGuess learning took up to 5h and its parameters were stored on hard disk, cf. Table 1. At query time, we employed a greedy gradient ascent algorithm for learning the topic 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: 2 Intel Xeon CPUs at 2.33GHz, 16 GB memory assigned to the JVM, and a RAID10 with IBM SAS $10K$ rpm disks. Before each query execution we cleared OS caches. Presented values are averages over five runs.

4.2 Selectivity Estimation Effectiveness

We employ the *multiplicative error* metric (*me*) [7] for measuring effectiveness:

$$me(Q) = \frac{\max\{\mathcal{F}_e(Q), \mathcal{F}_a(Q)\}}{\min\{\mathcal{F}_e(Q), \mathcal{F}_a(Q)\}}$$

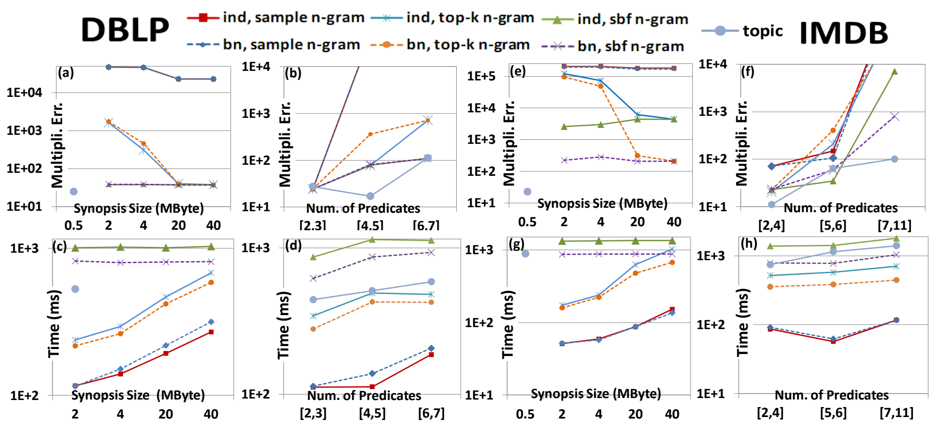


Fig. 4. Effectiveness: (a)+(b) for DBLP and (e)+(f) for IMDB. Efficiency: (c)+(d) for DBLP and (g)+(h) for IMDB. Y-axes are given in logarithmic scale.

with $\mathcal{F}_e(Q)$ and $\mathcal{F}_a(Q)$ as exact and approximated selectivity. Intuitively, $me(Q)$ is the factor to which $\mathcal{F}_a(Q)$ under-/overestimates $\mathcal{F}_e(Q)$.

Overall Results. Figs. 4-a/e (b/f) show the multiplicative error vs. synopsis size (# predicates) for DBLP and IMDB. Baseline system effectiveness strongly varies with their synopsis size. In particular, for small synopses $\leq$ 20 MB `ind` and `bn` performed poorly. We explain this with missing words in their string synopses, which led to heuristics being used for probability computation. In simple terms, `ind` and `bn` systems traded synopsis space for estimation accuracy.

TopGuess, on the other hand, did not suffer from this issue. All its parameters (cf. Sect. 3.1) 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 heuristic. We observed that TopGuess reduced the error of the best `bn` system, $\texttt{bn}_{\text{sbf}}$, by 88% (33%) for IMDB (DBLP). Further, we outperformed the best `ind` system, $\texttt{ind}_{\text{sbf}}$, by 99% (35%) on IMDB (DBLP).

Synopsis Size. Figs. 4-a/e show estimation errors w.r.t. in-memory synopsis size. An important observation is that the synopsis size is a key factor for effectiveness. Top-k and sample-based string synopsis systems were strongly affected by their (string) synopsis size. Given a small synopsis $\leq$ 4 MB, we observed that top-k/sample-based systems performed poorly. Here, many relevant query keywords were missed, leading to inaccurate heuristics being applied. With increasing synopsis size $\in [4, 20]$ MB, the performance of top-k approaches converged to the most accurate baseline (sbf-based systems). For instance given 4 MB space, the $\texttt{bn}_{\text{top-k}}$ approach preformed 95% worse than $\texttt{bn}_{\text{sbf}}$ on IMDB, but only 33% worse given 20 MB. Further, we noted sbf-based approaches to perform fairly stable. We explain this with sbf systems using bloom filters as an effective summary. Such systems were able to capture most query keywords. Thus, few heuristic-based estimations were necessary. However, sbf-based systems also have a limited memory space and must eventually omit words.

In contrast, we observed TopGuess to use $\leq$ 0.1 MB memory for IMDB as well DBLP. We explain this extremely compact BN with: (1) TopGuess has a network size, which is bound by the query size. (2) The BN only contains random variables that either are binary or have a sample space, which is bounded by the number of topics. For example, over the DBLP query load TopGuess needed on average 40 KB. Yet, TopGuess resolves the issue of missing words completely: the TopGuess parameters (stored on disk) capture all words in the vocabulary. At runtime, TopGuess retrieves the necessary statistics for a particular query and constructs its query-specific BN. This way, TopGuess achieved up to by 88% (33%) better results on IMDB (DBLP) than the best baselines.

Overall, we can conclude that estimation effectiveness is driven by accurate string predicate probabilities. Thus, there is a strong need for a data synopsis *allowing for extensive word/text data statistics.*

Correlations. We found system performances to vary for IMDB and DBLP. For the IMDB dataset, $\texttt{bn}_{\text{sbf}}$ could reduce errors of the $\texttt{ind}_{\text{sbf}}$ approach by up to 93%. 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 less correlations in DBLP than in IMDB. For instance, for DBLP queries with string predicates `name` and `label`, we noted no significant correlations. Thus, the probabilities obtained from `bn` systems were almost identical to the ones from `ind`.

In contrast, even for the less correlated dataset DBLP, TopGuess outperforms the best baselines, $\texttt{ind}_{\text{sbf}}$ and $\texttt{bn}_{\text{sbf}}$, by 35% and 33%. We explain this our a fine-grained, query-specific BN. More precisely, we observed that `bn` approaches exploited data correlations, which were observed in the data graph. However, TopGuess captured *even minor correlations via its topic mixtures at query time* – learned for each query individually.

Query Size. We depict the multiplicative error vs. # query predicates in Figs. 4-b/f. As expected, estimation errors increase for all systems in # predicates. For our baselines, we explain this behavior with: (1) Given an increasing # predicates, the chances of missing a query keyword increase. (2) When missing a single query keyword, the error is "propagated" throughout the computation.

However, while the TopGuess approach also led to more misestimates for larger queries, the degree of this increase was smaller. For instance, considering IMDB queries with 7-11 predicates, we could observe that TopGuess performs much more stable than `bn` or `ind` baselines, cf. Fig. 4-f.

4.3 Selectivity Estimation Efficiency

We now analyze the estimation efficiency vs. synopses size (# query predicates), cf. Figs. 4-c/g (d/h). For TopGuess, the reported times comprise parameter loading, BN construction, and topic learning. For `bn` and `ind`, the times represent only selectivity computation, i.e., no model learning or parameter loading.

Overall Results. Considering `bn` and `ind` systems, we saw that their string synopsis was a key performance factor. Intuitively, the more words were missed, the "simpler" and the more efficient these systems became. However, such gains came at the expense of effectiveness: the fastest baseline system, $\texttt{ind}_{\text{sample}}$, also computed the least accurate selectivity estimates.

Comparing the two systems with the best effectiveness, TopGuess and $\texttt{bn}_{\text{sbf}}$, TopGuess led to a better performance by up to 45%. Unfortunately, in comparison to top-k systems, TopGuess resulted in a performance decrease of 40%. We explain these drawbacks with the time-consuming disk I/O, which was needed for loading the statistics. However, while `bn` and `ind` clearly outperformed TopGuess w.r.t. small synopses $\leq$ 4 MB, TopGuess results are comparable for synopses $\geq$ 20 MB. We expect such effects to be more drastic for "large" `bn`/`ind` synopses $\gg$ 100 MB. So, *TopGuess guarantees a much more "stable" behavior.*

Synopsis Size. Figs. 4-c/g show time vs. synopsis size. For the baselines, we saw a correlation between synopsis size and runtime behavior: While `bn` and `ind` reach a high efficiency for synopses $\leq$ 4 MB, their performance decreases rapidly for synopses $\geq$ 20 MB. We explain this effect with the larger CPDs, which led to longer probability estimation times. We observed sbf-based approaches to be less driven by their synopsis size. This is because their computational costs are mainly determined by bloom filters. In contrast, TopGuess did not suffer from this issue at all. That is, for a given query, TopGuess only loads/processes statistics necessary for that query. All others statistics are kept on disk.

Query Size. All systems had increasing estimation times in query size, cf. Figs. 4-d/h. This is because each additional query predicate translated to more probability computations. However, as TopGuess exploits a compact query-specific BN, we expected it's performance to be less influenced by query size. To confirm this, we compared the standard deviation of the estimation time w.r.t. a varying # query predicates. For instance, the standard deviations was 82.48 ms (213.48 ms) for TopGuess (`bn`). The low deviation for TopGuess indicates that its probability estimation times varied less than those from `bn` systems.

5 Related Work

For selectivity estimation on *structured data*, existing works exploited various data synopses: join samples [1], graph synopses [18], or graphical models [9,20,21]. In particular, those synopses have been applied for selectivity estimation of structured SPARQL/BGP queries on RDF data, e.g., [7,10,17,19].

In contrast to TopGuess, such synopses do not consider correlations in text data or between text and structured data. In fact, the only work capturing correlations w.r.t. both kinds of data is [21]. However, [21] suffers from effectiveness (I.1) and efficiency issues (I.2), cf. Sect. 1 – as discussed throughout the paper.

With regard to selectivity estimation on *text data*, language models and other machine learning techniques have been employed [5,11,13,22]. Some works aim at substring or fuzzy string matching [5,13]. Other approaches target "extraction" operators, e.g., dictionary-based operators [22]. However, such works do not consider correlations among multiple string predicates or correlations between string predicates and query predicates for structured data.

6 Conclusion

We proposed a holistic approach for selectivity estimation of hybrid queries (TopGuess). We showed space and time complexity bounds for TopGuess. Further, we conducted empirical studies on real-world data and achieved strong effectiveness improvements, while not requiring additional runtime.

References

1. S. Acharya, P. Gibbons, V. Poosala, and S. Ramaswamy. Join synopses for approximate query answering. In *SIGMOD*, 1999.
2. V. Bicer, T. Tran, Y. Ma, and R. Studer. TRM - Learning Dependencies between Text and Structure with Topical Relational Models. In *ISWC*, 2013.
3. D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.
4. J. Chang and D. Blei. Relational Topic Models for Document Networks. In *AIStats*, 2009.
5. S. Chaudhuri, V. Ganti, and L. Gravano. Selectivity Estimation for String Predicates: Overcoming the Underestimation Problem. In *ICDE*, 2004.
6. J. Coffman and A. C. Weaver. A framework for evaluating database keyword search strategies. In *CIKM*, 2010.
7. A. Deshpande, M. N. Garofalakis, and R. Rastogi. Independence is Good: Dependency-Based Histogram Synopses for High-Dimensional Data. In *SIGMOD*, 2001.
8. F. Doshi, K. Miller, J. V. Gael, and Y. W. Teh. Variational Inference for the Indian Buffet Process. *JMLR*, 5:137–144, 2009.
9. L. Getoor, B. Taskar, and D. Koller. Selectivity estimation using probabilistic models. In *SIGMOD*, 2001.
10. H. Huang and C. Liu. Estimating Selectivity for Joined RDF Triple Patterns. In *CIKM*, 2011.
11. L. Jin and C. Li. Selectivity Estimation for Fuzzy String Predicates in Large Data Sets. In *VLDB*, 2005.
12. D. Koller and N. Friedman. *Probabilistic graphical models*. MIT press, 2009.
13. H. Lee, R. T. Ng, and K. Shim. Extending Q-Grams to Estimate Selectivity of String Matching with Low Edit Distance. In *VLDB*, 2007.
14. Y. Liu, A. Niculescu-Mizil, and W. Gryc. Topic-link LDA: Joint Models of Topic and Author Community. In *ICML*, 2009.
15. Y. Luo, W. Wang, X. Lin, X. Zhou, J. Wang, and K. Li. SPARK2: Top-k Keyword Query in Relational Databases. *TKDE*, 23(12):1763–1780, 2011.
16. A. Maier and D. E. Simmen. DB2 optimization in support of full text search. *IEEE Data Eng. Bull.*, 24(4):3–6, 2001.
17. T. Neumann and G. Moerkotte. Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins. In *ICDE*, 2011.
18. J. Spiegel and N. Polyzotis. Graph-based synopses for relational selectivity estimation. In *SIGMOD*, 2006.
19. M. Stocker, A. Seaborne, A. Bernstein, C. Kiefer, and D. Reynolds. SPARQL Basic Graph Pattern Optimization Using Selectivity Estimation. In *WWW*, 2008.
20. K. Tzoumas, A. Deshpande, and C. S. Jensen. Lightweight Graphical Models for Selectivity Estimation Without Independence Assumptions. *PVLDB*, 2011.
21. A. Wagner, V. Bicer, and T. D. Tran. Selectivity estimation for hybrid queries over text-rich data graphs. In *EDBT*, 2013.
22. D. Z. Wang, L. Wei, Y. Li, F. Reiss, and S. Vaithyanathan. Selectivity estimation for extraction operators over text data. In *ICDE*, 2011.
23. L. Zhang et al. Multirelational Topic Models. In *ICDM*, 2009.

7 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 [15], while IMDB queries are taken from [6]. All queries are given in RDF NTriples notation.[4]

Table 2. 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 1.1. Queries for DBLP [15]

```
# @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" .
```

[4] `http://www.w3.org/TR/rdf-testcases/#ntriples`

```
# 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 .
?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 1.2. Queries for IMDB [6]

```


# @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" .
```