

Attribute and Object Selection Queries on Objects with Probabilistic Attributes[†]

RABIA NURAY-TURAN
DMITRI V. KALASHNIKOV
SHARAD MEHROTRA
YAMING YU
University of California, Irvine

Modern data processing techniques such as entity resolution, data cleaning, information extraction, and automated tagging often produce results consisting of objects whose attributes may contain uncertainty. This uncertainty is frequently captured in the form of a set of multiple mutually exclusive value choices for each uncertain attribute along with a measure of probability for alternative values. However, the lay end-user, as well as some end-applications, might not be able to interpret the results if outputted in such a form. Thus, the question is how to present such results to the user in practice, e.g., to support *attribute-value selection* and *object selection* queries the user might be interested in. Specifically, in this paper we study the problem of maximizing the quality of these selection queries on top of such a probabilistic representation. The quality is measured using the standard and commonly used set-based quality metrics. We formalize the problem and then develop efficient approaches that provide high quality answers for these queries. The comprehensive empirical evaluation over three different domains demonstrates the advantage of our approach over existing techniques.

Categories and Subject Descriptors: H.2.m [Database Management]: Miscellaneous—*Selection queries on probabilistic data*; H.2.4 [Database Management]: Systems—*Query processing*; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—*Selection process, Retrieval models*

General Terms: Algorithms, Experimentation, Measurement, Performance, Reliability

Additional Key Words and Phrases: Attribute Value Selection, Object Selection Query, Result Quality, F-measure, Probabilistic Data

1. INTRODUCTION

The need of modern applications to process and analyze large volumes of raw data, such as data collected from web sources, has brought about the creation of a wide variety of data processing techniques. Many recent approaches have considered using probabilistic models for such data processing tasks. Examples include information extraction using conditional random fields [Satpal and Sarawagi 2007], probabilistic techniques to entity resolution [Kalashnikov and Mehrotra 2006; Wick et al. 2008], topic models [Asuncion et al. 2008], automated tagging of text [Steyvers et al. 2004], and of images [Kalashnikov et al. 2011]. Such probabilistic models often result in objects with probabilistic attributes wherein each attribute is represented as a set of mutually exclusive alternatives, each with an explicit estimate of probability of

[†]This work was supported in part by NSF grant CNS-1118114, DARPA grant HR0011-11-C-0017 and a gift from Google.

being the true value of the attribute. Our goal in this paper is to explore effective approaches for attribute and object selection queries over such a probabilistic representation, where the queries are defined as:

- (1) *Attribute Value Selection.* The query is for a given object to list the values of its attributes. Such a query arises, for instance, when a user browsing through the image collection is interested in the set of attributes the publisher has associated with the image, e.g., “25th anniversary celebration”.¹
- (2) *Object Selection Query.* Given query $Q = \{q_1, \dots, q_m\}$ the goal is to select all the objects whose attributes include query terms $\{q_1, \dots, q_m\}$.

Given this probabilistic representation and the two types of queries of interest, a natural question is: *What should we return as an answer to the end user?*

Observe that if the ground truth values were known for all uncertain attributes, then the database would have been deterministic and finding the correct ground truth answers G to these selection queries would have been straightforward. However, since the true values are not known, G is also not known, and any algorithm will need to *decide* how to choose its answer A , based on the possible values of object attributes and their probabilities.

One common solution to support these queries is to store the data in a probabilistic DBMS and then use the DBMS’s functionality to process the queries [Widom 2005; Antova et al. 2008; Singh et al. 2008; Dalvi and Suciu 2004]. The result of a selection query is an uncertain relation with tuples associated with the corresponding probabilities of being in the answer. However, from the perspective of the lay end-user, such a probabilistic output might not always be satisfactory. The main drawback is that a probabilistic result could be a long and overwhelming list of alternatives with associated probability values, which the end-user might not know how to interpret.

This drawback has indeed been long realized in previous research. As a result, various basic mechanisms to convert the probabilistic answer into the corresponding shorter and/or non-probabilistic result A have been considered (c.f. Section 2.5). Answer A computed by such techniques might not be equal to the true correct answer G , since G is unknown. The main challenge is thus to design an approach that computes *high-quality* results in a *computationally efficient* manner, where the quality is measured using some quality metric $M(A, G)$ that reflects the closeness of the result A to the true answer G .

In this paper we propose algorithms that maximize the quality for these two types of selection queries, where the quality is measured using the standard F_α -measure. F_α is a commonly used set-based quality metric that we will cover in more detail in Section 2.4. The proposed solution is based on estimating the expected quality of different answers and then picking the final answer as the one that maximizes the expected quality. While we develop our approach for the specific aforementioned probabilistic representation and queries, our techniques have the potential to benefit the probabilistic databases in general. Specially, the **main contributions** of this work are:

¹Attributes are essentially used to enhance the user’s experience of the image as in common social networking websites such as Facebook.

- A novel algorithm for attribute value selection. The algorithm has linear time complexity $O(n)$, where n is the number of alternative values for a given attribute. The proposed solution reaches optimal expected quality and outperforms the existing solutions under F_α quality metric (Section 3).
- An efficiency optimization technique for attribute selection query (Section 3.4)
- A novel algorithm for object selection query. The algorithm has linear time complexity $O(n)$, where n is the number of objects whose probability to satisfy the query is in $(0, 1]$ interval. In empirical studies, the proposed method reaches nearly optimal expected quality under F_α quality metric (Section 4).
- Efficiency optimization methods for object selection query (Section 4.7).
- An explanation of how the knowledge of dependencies and correlations in attribute data, when available, could be exploited to further increase the quality of the output for object selection queries (Section 4.8).
- Extensive empirical evaluation of the proposed solution (Section 5).

The rest of this paper is organized as follows. Preliminary material is covered in Section 2. Sections 3 and 4 provide solutions to attribute-value and object selection queries. These solutions are empirically evaluated in Section 5. Then the related work is covered in Section 6. Finally, Section 7 concludes the paper by highlighting the impact of the proposed approaches and outlining future work directions.

2. PRELIMINARIES

In this section we cover some preliminary basic material needed to understand the remainder of the paper. We first present motivating applications in Section 2.1. We then introduce our running example in Section 2.2. After that we introduce the notation and define the problem in Section 2.3. The base quality metrics that will be used in the paper are then covered in Section 2.4. We present the baseline solutions for the problem in Section 2.5. Finally, we outline our overall approach in Section 2.6.

2.1 Motivating Applications

We motivate the usefulness and applicability of the proposed query answering approach by drawing upon three data processing techniques that produce data with probabilistic attributes, which are overviewed next.

Entity Resolution. The first one is an instance of the entity resolution problem known as fuzzy lookup [Chaudhuri et al. 2003; Kalashnikov and Mehrotra 2006; Nuray-Turan et al. 2007; Chen et al. 2007]. In the setting of this problem the algorithm is given a dataset that contains objects with uncertain attributes \mathcal{O}_{drt} and a set of clean objects \mathcal{O}_{cln} . Each uncertain attribute is a reference to one object from \mathcal{O}_{cln} , but the description provided by the reference can be ambiguous and match multiple objects from \mathcal{O}_{cln} . The task is to find for each uncertain reference r the right object $O \in \mathcal{O}_{cln}$ that r refers to. For instance, in a publication database scenario, \mathcal{O}_{drt} can be a a publication table storing, among other publication attributes, uncertain references to authors, e.g. “J. Smith”. Set \mathcal{O}_{cln} can be a clean author table storing complete information on all possible authors. The task can be to determine for each uncertain author reference which specific author in the author table it refers to. The output of fuzzy lookup is a set of possible authors along with

the corresponding probabilities that the uncertain attribute refers to these authors [Kalashnikov and Mehrotra 2006; Kalashnikov et al. 2005; Chen et al. 2009].

Information Extraction. Information extraction and slot filling is the second class of data processing techniques that can use the probabilistic representation, e.g. [Ashish et al. 2009; Satpal and Sarawagi 2007]. For example, the task might be to extract from a corpus of web pages information about people mentioned there, including their affiliations. An extractor would often associate mutually exclusive alternatives with each possible affiliation along with its confidence in each alternative.² These confidences can be converted into the corresponding probabilities.

Speech Tagging. The third data processing technique that can utilize the probabilistic representation are various content annotation and speech tagging approaches that employ speech recognition, e.g. [Chen et al. 2001; Kalashnikov et al. 2011]. For instance, most camera devices have built-in microphones and provide mechanisms to associate images with speech input. The user would take a picture and speak the desired attributes into the device’s microphone. A speech recognizer transcribes the audio signal into text. This text is used in assigning attributes to the image. Such annotation approaches have been demonstrated to be useful in variety of application settings, including reconnaissance and crisis response [Kalashnikov et al. 2011; Desai et al. 2009]. When processing an utterance of a word a speech recognizer typically associates the N-best list for the utterance. The N-best list contains up to N mutually exclusive alternatives, in the form of text, for the utterance, along with the corresponding confidences. These confidences are then transformed into the probabilities. For example, for the utterance of a word “fire” the N-best list, where N is 3, might be “flyer”, “fire”, and “fare” with the probabilities 0.6, 0.3, and 0.1. We will use the image tagging scenario to derive most of the illustrative examples in this paper.

2.2 Running Example

Let us introduce a running example of an object with probabilistic attributes. The example will be used to better illustrate various concepts covered in the paper.

Figure 1 illustrates the probabilistic representation in the context of automated image tagging. It shows an image annotated with three tags that correspond to attributes a_1 , a_2 , and a_3 . The first tag a_1 has two possible options, it can be either *sun* with probability 0.6 or *fun* with probability 0.4. The second tag a_2 can be either *peach*, *beach*, or *reach* with probabilities 0.4, 0.3, and 0.3. The options for the third tag a_3 are *tree*, *three*, and *free* with probabilities 0.7, 0.2, and 0.1 respectively. The ground truth values for these attributes are *sun*, *beach*, and *tree*, they are marked bold in the figure.

Assume that we want to represent the information about image tags in a Trio-like probabilistic DBMS [Widom 2005]. Then it can be achieved, for instance, by creating **Tags** table that would store the association between tags and images, as well as tag values in the form of $\langle \text{Obj_ID}, \text{Tag_ID}, \text{Tag_Val} \rangle$ tuples, as illustrated in Table I.³ Here, **Tag_Val** is the uncertain attribute that stores multiple mutually

²If an extractor has to commit to just one value for the affiliation, it would typically pick the top most likely alternative.

³This is just a simplified example to illustrate attribute and object selection queries. It does not

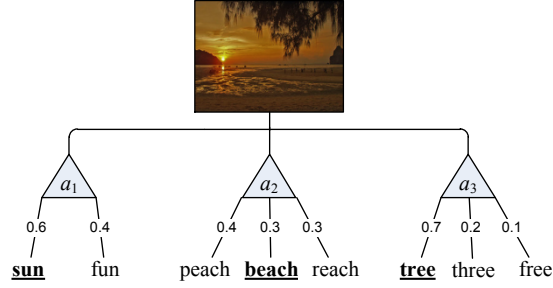


Fig. 1. Example of Uncertain Attributes.

Obj_ID	Tag_ID	Tag_Val
1	1	sun 0.6; fun 0.4
1	2	peach 0.4; beach 0.3; reach 0.3
1	3	tree 0.7; three 0.2; free 0.1

Table I. Tags table. Showing one annotated image.

exclusive values.

Then, an example of an attribute value selection query where the task is to list all tags of object with id of 100, would be:

```
SELECT Tag_ID, Tag_Val
FROM Tags
WHERE Img_ID = 100
ORDER BY Tag_ID;
```

Similarly, an example of object retrieval query, where the goal is to select all objects that contain tag 'sun', would be:

```
SELECT DISTINCT Obj_ID
FROM Tags
WHERE Tag_Val = 'sun';
```

To get objects annotated with both 'sun' and 'beach', an object retrieval query involves a join and would be:

```
SELECT DISTINCT T1.Obj_ID
FROM Tags T1, Tags T2
WHERE T1.Obj_ID = T2.Obj_ID AND
      T1.Tag_Val = 'sun' AND
      T2.Tag_Val = 'beach';
```

Notice that the above queries address a database problem of finding images that have certain tags. They are not solving a computer vision problem of finding images

reflect all the capabilities of modern probabilistic DBMS's.

Notation	Meaning
\mathcal{D}	the dataset
O_j	j -th object in \mathcal{D}
$\mathcal{O} = \{O_1, O_2, \dots, O_{ \mathcal{O} }\}$	the set of all objects in \mathcal{D}
a	an uncertain attribute of some object
g_a	the unknown ground truth value of a
v_i	i -th possible value of a
$p_i = \mathbb{P}(v_i = g_a)$	probability that v_i is the right choice
$V = \{v_1, v_2, \dots, v_n\}$	set of possible values of a
$P = \{p_1, p_2, \dots, p_n\}$	probabilities for V , $p_1 \geq \dots \geq p_n$
$w_i \in [0, 1]$	algorithm's confidence that v_i is g_a
$W = \{w_1, w_2, \dots, w_n\}$	confidence vector for a
G_Q	ground truth answer for query Q
$p_i = \mathbb{P}(O_i \in G_Q)$	probability that O_i belongs to G_Q

Table II. Notation

that depict objects described by the query terms. That is, for a query ‘sun’ the task is to find images that have a tag ‘sun’ and not images that contain sun in them. The latter is a large area of research, but not the focus of this paper.

2.3 Notation and Problem Definition

After applying one of the applications described in Section 2.1, the resulting dataset \mathcal{D} will consist of, among other things, a set of objects $\mathcal{O} = \{O_1, O_2, \dots, O_{|\mathcal{O}|}\}$. Each object O_i is annotated with a set of k_i attributes $\{a_{i1}, a_{i2}, \dots, a_{ik_i}\}$ as shown in Figure 2. The ground truth value g_a of each attribute a is uncertain and is not known in general. Instead, each attribute a is given as a random variable that takes value $v_i \in V_a$ with probability $p_i \in P_a$. The *option set* $V_a = \{v_1, v_2, \dots, v_{n_a}\}$ contains mutually exclusive possible values for a and the *probability set* $P_a = \{p_1, p_2, \dots, p_{n_a}\}$ stores the corresponding probabilities, where $p_i = \mathbb{P}(v_i = g_a)$ and $\sum_i p_i = 1$. For notational convenience, for each attribute a we will renumber the v_i 's such that $p_1 \geq p_2 \geq \dots \geq p_{n_a}$. Together, V_a and P_a encode the *probability mass function* (pmf) for a , which, unlike g , is known for each attribute a . The notation is summarized in Table II.

Example 2.3.1. Consider the running example in Figure 1. The dataset \mathcal{D} is the collection of images. The shown image corresponds to some object $O_j \in \mathcal{D}$ that has three uncertain attributes a_1 , a_2 , and a_3 . The possible values for attribute a_1 are $v_{11} = \text{‘sun’}$ and $v_{12} = \text{‘fun’}$, the corresponding probabilities for these values are $p_{11} = 0.6$ and $p_{12} = 0.4$. The ground truth value for a_1 is $g_1 = \text{‘sun’}$. Similarly, for attribute a_2 we have $v_{21} = \text{‘peach’}$, $v_{22} = \text{‘beach’}$, and $v_{23} = \text{‘reach’}$, with probabilities $p_{21} = 0.4$, $p_{22} = 0.3$, and $p_{23} = 0.3$, and ground truth $g_2 = \text{‘beach’}$. \square

Let us consider attribute value selection query on attribute a_1 for the running example. The possible set-based answers to the query could be $A_1 = \{\}$, $A_2 = \{\text{sun}\}$, $A_3 = \{\text{fun}\}$, and $A_4 = \{\text{sun}, \text{fun}\}$.⁴ Only one of these answers, A_2 , is equal to the ground truth answer $G = \{\text{sun}\}$. The remaining answers are either

⁴In Section 3 we will consider a more general answer representation that allows to associate confidences with each element in the answer set.

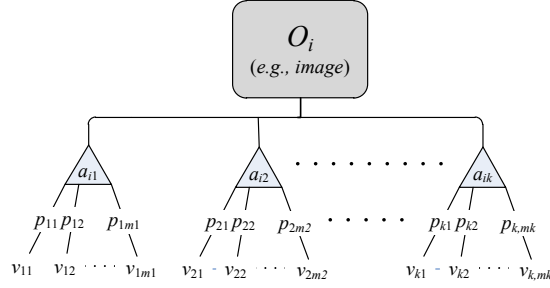


Fig. 2. Uncertain Object Attributes.

inaccurate (they do not contain *sun*) or only partially accurate (they contain wrong answers in addition to *sun*). It is possible to use a quality metric $M(A_i, G)$, such as F_α -measure explained in Section 2.4, for assessing the quality of answers.⁵ An approach will need to pick one A_i that it thinks will most likely result in the highest quality. Furthermore, given that there could be many possible answers, it will need to pick its answer quickly to be feasible in practice.

The overall goal is to design *efficient* and *high-quality* approaches for answering the two types of selection queries on top of this probabilistic representation. This goal will be formalized specifically for attribute and object selection queries in Sections 3 and 4 respectively.

2.4 Base Quality Metrics

We need to be able to assess the goodness of possible answers produced by various algorithms for attribute and object selection queries. Given that we are dealing with set-based answers, we can use one of the standard set-based metrics, such as Dice, Jaccard, or F_α -measure [Baeza-Yates and Riberto-Neto 1999]. We will primarily focus on F_α metric, commonly used in database literature.

Given query Q , let G be the ground truth answer set for Q , and A be the answer set produced by an algorithm. Then these generic metrics are defined as [Baeza-Yates and Riberto-Neto 1999]:

$$Precision(A, Q) = \frac{|A \cap G|}{|A|}, \quad (1)$$

$$Recall(A, Q) = \frac{|A \cap G|}{|G|}, \quad (2)$$

$$F_\alpha(A, Q) = \frac{(1 + \alpha) \cdot Precision(A, Q) \cdot Recall(A, Q)}{\alpha \cdot Precision(A, Q) + Recall(A, Q)}. \quad (3)$$

The purpose of precision is to assess the *purity* of the answer. By its definition, precision measures the fraction of the correct items in the answer. The purpose of recall is to assess the *completeness* of the answer. It measures the fraction of the correct elements in the answer.

⁵Under F_α , the quality of answer $A = \{\}$ will always be zero for attribute retrieval, so this answer can be omitted from consideration.

Frequently, a tradeoff exists between precision and recall. That is, it is often easy to get answers with high precision but low recall and vice versa.⁶ Since it is desirable to get answers that have both high precision and recall at the same time, these two metrics are often combined into a single F_α measure as defined in (3). Parameter α controls the weight of precision. It is set by the domain expert depending on a particular application, since some applications prefer higher recall over precision, whereas others require the reverse. One commonly used value of α is 1 where contribution of precision and recall are equal and the measure is referred to as F_1 -measure.

Example 2.4.1. Assume that the correct (ground truth) answer to query Q is $G = \{O_1, O_2, O_5, O_6, O_9, O_{10}\}$. Assume that an algorithm computes its answer to Q as $A = \{O_1, O_3, O_5, O_7, O_9\}$. Then the correct elements in A are $A \cap G = \{O_1, O_5, O_9\}$ and thus the precision of A is $\frac{3}{5}$ as only 3 out of 5 elements are correct. Similarly, the recall of A is $\frac{3}{6} = \frac{1}{2}$ since only 3 out of 6 correct answers are returned in A . \square

2.5 Basic Solutions

It is possible to use existing techniques to develop basic solutions to the problem of answering attribute and object selection queries on top of the probabilistic representation. These techniques are reviewed next.

2.5.1 Attribute Selection Techniques. The techniques listed below have prefix **Attr-** to highlight the fact that they are applied to attribute values:

Attr-Top-1. For an attribute a , among $\{v_1, v_2, \dots, v_{n_a}\}$ values, the Top-1 approach, picks the top-1 option v_1 as having the highest probability p_1 of being the correct value for a . The object O_i is thus annotated with the most likely correct values of its attributes. This approach can also be used for object selection queries, where O_i is selected when its top-1 attribute values contain the query terms $\{q_1, q_2, \dots, q_m\}$.

Attr-Top-K. **Attr-Top-1** approach can be generalized into **Attr-Top-K** solution, which picks the top K values from V to be associated with attribute a . If $|V| < K$ then **Attr-Top-K** will associate with a all values from V .

Attr-All. An alternative to **Attr-Top-1** is **Attr-All** approach that associates with each attribute a all of its possibilities $\{v_1, v_2, \dots, v_{n_a}\}$. In terms of object selection queries, **Attr-All** always leads to the best possible recall since it uses all possible values. The precision however will suffer, as many irrelevant attributes are associated with each object O_i , leading to a possible wrongful selection of O_i when these attributes are used in a query.

Attr-Thr- τ . Thresholding-based approach is the middle ground between **Attr-Top-1** and **Attr-All** strategies. It employs a pre-specified cut-off thresholds τ and associates with each attribute a the subset of $\{v_1, v_2, \dots, v_{n_a}\}$ consisting of v_1 and all the v_i 's such that $p_i \geq \tau$.

2.5.2 Object Selection Techniques. For object queries, one solution is first to fix the attributes using one of the above strategies. This essentially will create

⁶For instance, including all elements of the dataset in the answer will lead to the highest possible recall, but likely at the cost of poor precision.

a deterministic/non-probabilistic version of the dataset. Then the traditional object selection can be employed. We will refer to these approaches as **Attr-Top-1**, **Attr-Top-K**, **Attr-All**, **Attr-Thr- τ** .

We can observe that **Attr-Top-1** is an excellent choice for applications that *require* to commit to a single value for each uncertain attribute, for both attribute and object selection queries. However, the important question that we study in this paper is whether higher-quality approaches can be designed for many applications that do not have such a requirement. For instance, we can notice that this approach may fail to include in the answer set many relevant objects wherein the ground truth values of the attributes happened not to get the highest probability. Thus, the **Top-1** approach might be suboptimal for applications that give preference to high recall.

Another approach is **Thr- τ** method that selects the objects whose probabilities to belong to the ground truth exceed the predefined threshold τ . A way to compute these probabilities will be discussed in Section 4.

Example 2.5.1. Let us consider the example illustrated in Figure 1. **Attr-Top-1** approach will pick *sun*, *peach*, and *tree* as values for tags a_1 , a_2 , and a_3 and thus it will incorrectly assign tag a_2 whose correct value is *beach*. **Attr-All** approach will associate all the possibilities with each tag and thus the image will be wrongfully returned if the query is say $Q = \{fun\}$, since *fun* is one of the possible options for tag a_1 . **Threshold** approach for $\tau = 0.4$ will pick *sun* and *fun* for a_1 , *peach* for a_2 , and *tree* for a_3 and thus the image also will be wrongfully returned for, say, $Q = \{fun\}$. \square

2.6 Overview of our MoE Approach

Our goal is to efficiently compute answer A to selection queries that would maximize quality $M(A, G)$, where we fix the quality metric to be F_α -measure. One of the challenges is that the ground truth answer G is unknown to the algorithm beforehand, so that it cannot measure quality $M(A, G)$ of different answers directly.

The idea of our solution, to which we refer as Maximization of Expectation (MoE), is to find answer A_{opt} that would maximize the quality in the *expected* sense. Let \mathcal{G} be the space of all possible ground truth answers, and $G_i \in \mathcal{G}$ be one particular possible ground truth. Then, for a given answer A we can compute its expected quality:

$$\mathbb{E}(F_\alpha(A)) = \sum_{G_i \in \mathcal{G}} F_\alpha(A, G_i) \mathbb{P}(G_i = G).$$

Let \mathcal{A} be the space of all feasible answers. The best answer then can be chosen as:

$$A_{opt} = \operatorname{argmax}_{A \in \mathcal{A}} \mathbb{E}(F_\alpha(A)).$$

The intuition is that while A_{opt} might not be equal to G , it is the best answer to output given that G could be any of G_i with probabilities $\mathbb{P}(G_i = G)$. Computing A_{opt} however can be quite costly, thus the challenge that needs to be addressed is how to compute it in an efficient manner. We show that a linear time algorithm exists for attribute values selection. For object selection queries, we show how these expected values can be *estimated* quickly, which leads to an algorithm that reaches high quality and at the same time has linear time complexity.

3. ATTRIBUTE VALUE SELECTION

In this section we study the problem of attribute value selection. Most of the discussion will be presented in the context of a single attribute and the corresponding F-measure. For an object with multiple attributes, the overall F-measure is computed as the average F-measure over all of the object’s attributes. We start by discussing how to choose a quality metric and answer format in Section 3.1. We then define the expected quality measures in Section 3.2. With the help of these measures we show that using the proposed confidence-based representation reduces to the case of set-based answers only, and prove a theorem for assigning optimal (in the expected sense) values to an attribute in Section 3.3. We then introduce an efficiency optimization of the algorithm in Section 3.4.

3.1 Choosing Quality Metric and Answer Format

Given attribute a , its $V = \{v_1, v_2, \dots, v_n\}$ and $P = \{p_1, p_2, \dots, p_n\}$ and the fact that a has a single unknown ground truth value g the task is to decide how to present the algorithm’s answer for a to the end user. To do so we first need to choose the *output format* to represent such results and then develop a method for *measuring the quality* of the outputted answer. We are not aware of any specific prior work that addresses these questions directly for the problem of attribute selection.

3.1.1 Set-Based Answer Format. For some applications there could be a restriction placed on the format of the answer that would require that a single value v_i from V be chosen as the answer for attribute a . Then the Top-1 approach is the right solution. However, if there is no such requirement, such as in the image tagging context, then a subset A of V , consisting of multiple elements, can be outputted as value of a . This set-based answer format has the benefit of showing to the user a set of elements in which (s)he can recognize (or be informed of) the correct one. However, this set should be chosen carefully such that it has the high chance of containing the correct answer while at the same time it should not overwhelm the user with too many wrong answers. Specifically, the algorithm can represent its answer A in the form of a confidence vector $B = \{b_1, b_2, \dots, b_n\}$, where binary value $b_i \in \{0, 1\}$ is set to $b_i = 1$ if the algorithm decides to put v_i in A , and $b_i = 0$ if $v_i \notin A$.

Example 3.1.1. Consider a case where attribute a has 12 possible values $V = \{v_1, v_2, \dots, v_{12}\}$, where $p_1 = 0.5$, $p_2 = 0.4$, and $p_3 = p_4 = \dots = p_{12} = 0.01$. Then examples of answers and their encodings are:

- (1) $A_1 = \{v_1\}$ as $B_1 = \{1, 0, 0, \dots, 0\}$,
- (2) $A_2 = \{v_2\}$ as $B_2 = \{0, 1, 0, \dots, 0\}$,
- (3) $A_{1,2} = \{v_1, v_2\}$ as $B_{1,2} = \{1, 1, 0, 0, \dots, 0\}$,
- (4) $A_{1,2,4} = \{v_1, v_2, v_4\}$ as $B_{1,2,4} = \{1, 1, 0, 1, 0, 0, \dots, 0\}$, and
- (5) $A_{1,2,\dots,12} = \{v_1, v_2, \dots, v_{12}\}$ as $B_{1,2,\dots,12} = \{1, 1, \dots, 1\}$. □

3.1.2 Set-Based Quality Metric. If the ground truth answer for attribute a is $G = \{g\}$, whereas the algorithm outputs its answer as a set $A \subseteq V$, we need to define what the utility $U[A, g]$ of the set-based answer to the user is. While the question of utility, or quality, of answer have been explored for various types of

queries in the past [Baeza-Yates and Riberto-Neto 1999], it has not been actively studied for attribute selection queries, so different solutions are possible. Intuitively we want a utility function to satisfy the following requirements:

- (1) $0 \leq U[A, g] \leq 1$. The utility function is normalized to $[0,1]$ interval.
- (2) $U[A, g] = 0$ iff $g \notin A$. If answer A does not contain the ground truth element g then its utility is zero, otherwise it is above zero.
- (3) $U[A, g] = 1$ iff $A = \{g\}$. The utility reaches 1 if and only if the answer is equal to the ground truth element g .
- (4) $0 < U[A, g] < 1$, if $g \in A$ and $|A| \geq 2$. This follows from (1), (2) and (3).
- (5) if $g \in A_1, g \in A_2, |A_1| < |A_2|$ then $U[A_1, g] > U[A_2, g]$. If two answers contain the ground truth value, but one is shorter, then it should be preferred so as not to overwhelm the user.

While there could be several ways to chose the utility function, in this paper we study set-based quality metrics. Specifically we will focus on the standard F_α measure which has been explained in Section 2.4. Other common set-based metrics such as Dice and Jaccard will satisfy the above requirements and are expected to prefer similar results as F_1 , that is, F_α where α is set to its standard value of 1. F_α measures how much the outputted answer $A \subseteq V$ is different from the ground truth answer $G = \{g\}$. More formally, for an answer A encoded with the help of binary confidence vector B , from Eqs. (1), (2), and (3) we get:

$$Precision(B, a) = \frac{\sum_{i=1}^n d_i b_i}{\sum_{i=1}^n b_i} = \frac{b_g}{\sum_{i=1}^n b_i}, \quad (4)$$

$$Recall(B, a) = \frac{\sum_{i=1}^n d_i b_i}{1} = \sum_{i=1}^n d_i b_i = b_g, \quad (5)$$

$$F_\alpha(B, a) = \frac{(1 + \alpha) \frac{b_g}{\sum_{i=1}^n b_i} b_g}{\alpha \frac{b_g}{\sum_{i=1}^n b_i} + b_g} = \frac{(1 + \alpha) b_g}{\alpha + \sum_{i=1}^n b_i}. \quad (6)$$

Here d_i is the indicator of the ground truth value: $d_i = 1$ if $v_i = g$ and $d_i = 0$ if $v_i \neq g$. Variable $b_g \in \{0, 1\}$ is the binary weight assigned to the ground truth value g , that is, $b_g = b_j$ s.t. $b_j = g$. It is trivial to check that F_α satisfies all of the aforementioned requirements.

Example 3.1.2. Consider the case from Example 3.1.1. Assume that the ground truth value for a is $g = v_2$. Then

- (1) $F_1(A_1) = \frac{(1+1) \cdot 0}{1+1} = 0$,
- (2) $F_1(A_2) = 1$,
- (3) $F_1(A_{1,2}) = \frac{(1+1) \cdot 1}{1+2} = \frac{2}{3}$,
- (4) $F_1(A_{1,2,4}) = \frac{(1+1) \cdot 1}{1+3} = \frac{2}{4}$, and
- (5) $F_1(A_{1,2,\dots,12}) = \frac{(1+1) \cdot 1}{1+12} = \frac{2}{13}$.

Observe how F_1 -measure prefers shorter answers that contain the ground truth element v_2 . \square

Remark 1. It is interesting to study the connection between F_1 and other two common set-based metrics Jaccard and Dice for our case. Let $k = |A|$, then from Eq. (6) we get $F_1(B) = 2 \frac{b_g}{1+k}$. For Jaccard we get $J(A, G) = \frac{|A \cap G|}{|A \cup G|} = \frac{b_g}{k}$. For Dice, $D(A, G) = \frac{2|A \cap G|}{|G| + |A|} = 2 \frac{b_g}{1+k} = F_1(B)$. Thus, in this case Dice is identical to F_1 . \square

3.1.3 Set-Based Metrics with Confidences. Some users, as well as certain applications, in addition to set-based answer A for a might also want to see confidences w_i be associated with each element $v_i \in A$. More generally, the idea of representing an answer with a binary membership vector $B = \{b_1, b_2, \dots, b_n\}$ can be generalized to representing the answer with a confidence vector $W = \{w_1, w_2, \dots, w_n\}$, where $w_i : 0 \leq w_i \leq 1$ is a real-valued confidence/weight that the algorithm should associate with each possible value $v_i \in V$. This weight should represent the degree of confidence of the algorithm that v_i should be included in the answer set A . Specifically, we will consider the semantics where w_i represents the degree of membership of v_i in A . For example, $w_i = 0$ / $w_i = 0.5$ / $w_i = 1$ mean the algorithm associates zero/half/full mass of v_i with A .

Similar to the case of set-based answers, we can define the notion of the utility $U'[A, g]$ of a confidence-based answer A to the user. Some natural requirements for $U'[A, g]$ are similar to those of $U[A, g]$, but now they are formulated in terms of weights. Let $w_A = \sum_{i=1}^n w_i$ be the weight of answer A , then:

- (*) If $w_i \in \{0, 1\}$ for all i , then $U'[A, g] \equiv U[A, g]$. If the chosen weights happen to be only 0 or 1, then confidence-based answer is equivalent to set based answer and $U'[A, g]$ should behave the same way as $U[A, g]$.
- (1') $0 \leq U'[A, g] \leq 1$. The utility function is normalized to $[0, 1]$ interval.
- (2') $U'[A, g] = 0$ iff $w_g = 0$. If the weight w_g assigned to g in this answer is zero then the utility of the answer is also zero.
- (3') $U'[A, g] = 1$ iff $w_A = w_g = 1$. The utility reaches 1 if and only if the ground truth element g gets weight of 1 (i.e., $w_g = 1$) and all other elements get weight of zero (i.e., $w_A = w_g$).
- (4') $0 < U'[A, g] < 1$, if $w_A > w_g > 0$. Follows from (1'), (2') and (3').
- (5') For A_1 and A_2 , if $w_g^1 = w_g^2 > 0$ but $w_{A_1} < w_{A_2}$ then $U'[A_1, g] > U'[A_2, g]$. If two answers assign the same positive weight to g (i.e., $w_g^1 = w_g^2 > 0$), but one answer assigns less weight to incorrect elements (and thus, $w_{A_1} < w_{A_2}$) then the utility of that answer is larger.

While there could be various metrics that satisfy these conditions, we will use weighted versions of F-measure, which is a harmonic mean of weighted precision, recall. These metrics substitute boolean b_i 's with real valued weights w_i 's:

$$Precision(W, a) = \frac{w_g}{\sum_{i=1}^n w_i}, Recall(W, a) = w_g, F_\alpha(W, a) = \frac{(1 + \alpha)w_g}{\alpha + \sum_{i=1}^n w_i}. \quad (7)$$

Here, similar to b_g , variable w_g is the weight assigned to the ground truth value g , that is $w_g = w_j$ s.t. $v_j = g$. Similar metrics have been used in the context of

information extraction, speech segmentation, and information retrieval to measure quality when the degree of membership of an element to a set is expressed as a real value [Carroll and Briscoe 2002; Martín-Bautista et al. 2000; Ziolkó et al. 2007; Ravindra et al. 2004].

Observe that using confidence vectors provides great flexibility for an algorithm to output its answers. If the algorithm is not fully confident in whether some value v_i should be in the answer set for attribute a , it can output its confidence weight w_i that is less than 1, such as $w_i = 0.8$. The metrics provided in Eq. (7) are capable of factoring in confidences, as they encourage the algorithm to give greater weight to the right option and discourage it from doing so to the wrong options.

Example 3.1.3. Let us consider some of the possible answers:

- (1) $A_{v_1:1, v_2:1}$ with $W = \{1, 1, 0, 0, \dots, 0\}$, is where v_1 and v_2 are fully included in the answer – it is identical to $A_{1,2}$ above.
- (2) $A_{v_1:0.5, v_2:0.4}$ with $W = \{0.5, 0.4, 0, 0, \dots, 0\}$ is where only v_1 and v_2 are in the answer and $w_1 = p_1$ and $w_2 = p_2$.
- (3) $A_{v_1:\frac{5}{9}, v_2:\frac{4}{9}}$ with $W = \{\frac{5}{9}, \frac{4}{9}, 0, 0, \dots, 0\}$ is where only v_1 and v_2 are in the answer and w_1, w_2 are “normalized” probabilities $w_1 = \frac{p_1}{p_1+p_2} = \frac{5}{9}$ and $w_2 = \frac{p_2}{p_1+p_2} = \frac{4}{9}$, chosen such that they are proportional to p_1 and p_2 and $w_1 + w_2 = 1$.
- (4) $A_{1,2,\dots,12}$ with $W = \{0.5, 0.4, 0.01, 0.01, \dots, 0.01\}$ is the answer with all the elements from V and where $w_i = p_i$ for $i = 1, 2, \dots, 12$. It represents the case where the weights are set to the corresponding probabilities.

Then we have:

- (1) $F_1(A_{v_1:1, v_2:1}) = F_1(A_{1,2}) = \frac{2}{3} \approx 0.67$,
- (2) $F_1(A_{v_1:0.5, v_2:0.4}) = \frac{(1+1) \cdot 0.4}{1+0.9} \approx 0.42$,
- (3) $F_1(A_{v_1:\frac{5}{9}, v_2:\frac{4}{9}}) = \frac{(1+1) \cdot \frac{4}{9}}{1+1} \approx 0.44$, and
- (4) $F_1(A_{1,2,\dots,12}) = \frac{(1+1) \cdot 0.4}{1+1} = 0.4$.

Observe that the weighted F_1 -measure prefers the first answer $A_{v_1:1, v_2:1}$: it is short and places high confidence of 1 into the correct element v_2 . The second and third answers are also short, but the placed confidences in the right element v_2 are much lower, so these answers are scored lower by F_1 . While the combined weight of elements in the fourth answer is only 1, unlike the first answer, it places much lower confidence in v_2 so its score is also lower. \square

Remark 2. Let us compare weighted versions of F_1 , Jaccard, and Dice for our case. From Eq. (7) we have $F_1(A) = \frac{2w_g}{1+w_A}$. For Jaccard and Dice we have $J(A, G) = \frac{w_g}{w_A}$ and $D(A, G) = \frac{2w_g}{1+w_A} = F_1(A)$. Hence, in this case Dice is equivalent to F_1 and all the proofs in this paper will hold if F_α is replaced with Dice. \square

For weighted set-based metrics the overall task thus becomes of finding the optimal combination of weights W that maximizes the user’s quality metric. In the next sections we will prove a base theorem that if the user chooses the weighted F_α defined in Eq. (7) as the quality metric, then it is sufficient to consider only binary weights as described in Section 3.1.2. Hence, there will be no need to “interpret”

real-valued confidences by the end user, as each $v_i \in V$ is either fully included, or not, in the answer. We will also show how to construct the optimal (in the expected sense) answers for attribute value selection queries.

3.2 Expected Measures

Since weighted F_α measure in Eq. (7) is a generalization of F_α measure in Eq. (6), in the subsequent discussion we will focus on the more general case of weighted F_α . To answer the question of which set of weights $W = \{w_1, w_2, \dots, w_n\}$ the algorithm should choose, for given V and P , to maximize the quality of answer, let us define the *expected* versions of precision, recall, and F_α . Observe that in Eq. (7) weight w_g can be one of w_1, w_2, \dots, w_n with the corresponding probabilities of p_1, p_2, \dots, p_n . Therefore, from Eq. (7) for a given answer W the expected value of precision is computed as the probabilistic average:

$$\mathbb{E}(\text{Precision}(W, a)) = \frac{\sum_{i=1}^n p_i w_i}{\sum_{i=1}^n w_i}. \quad (8)$$

Similarly, the expected recall is:

$$\mathbb{E}(\text{Recall}(W, a)) = \sum_{i=1}^n p_i w_i. \quad (9)$$

And, the expected value of F_α is:

$$\mathbb{E}(F_\alpha(W, a)) = \frac{(1 + \alpha) \sum_{i=1}^n p_i w_i}{\alpha + \sum_{i=1}^n w_i}. \quad (10)$$

Example 3.2.1. For the binary case from Example 3.1.2 we have:

- (1) $\mathbb{E}(F_1(A_1)) = \frac{(1+1) \cdot 0.5 \cdot 1}{1+1} = 0.5$,
- (2) $\mathbb{E}(F_1(A_2)) = \frac{(1+1) \cdot 0.4 \cdot 1}{1+1} = 0.4$,
- (3) $\mathbb{E}(F_1(A_{1,2})) = \frac{(1+1)(0.5 \cdot 1 + 0.4 \cdot 1)}{1+2} = 0.6$,
- (4) $\mathbb{E}(F_1(A_{1,2,4})) = \frac{(1+1)(0.5 \cdot 1 + 0.4 \cdot 1 + 0.01 \cdot 1)}{1+3} = 0.455$, and
- (5) $\mathbb{E}(F_1(A_{1,2,\dots,12})) = \frac{(1+1) \cdot (0.5 + 0.4 + \dots + 0.01)}{1+12} = \frac{2}{13} \approx 0.15$.

Among these answers expected F_α prefers $A_{1,2}$. This is not surprising since $A_{1,2}$ is short and has the high chance of 0.9 to include the correct answer, though it will always have either 1 or 2 incorrect elements. While answer $A_{1,2,\dots,12}$ has 100% chance of including the correct element, it is long, and the remaining 11 elements are incorrect. Let us now consider confidence-based F_α from Example 3.1.3:

- (1) $\mathbb{E}(F_1(A_{v_1:1, v_2:1})) = \mathbb{E}(F_1(A_{1,2})) = 0.6$,
- (2) $\mathbb{E}(F_1(A_{v_1:0.5, v_2:0.4})) = \frac{(1+1)(0.5 \cdot 0.5 + 0.4 \cdot 0.4)}{1+0.9} \approx 0.43$,
- (3) $\mathbb{E}(F_1(A_{v_1:\frac{5}{9}, v_2:\frac{4}{9}})) = \frac{(1+1)(0.5 \cdot \frac{5}{9} + 0.4 \cdot \frac{4}{9})}{1+1} \approx 0.46$, and
- (4) $\mathbb{E}(F_1(A_{1,2,\dots,12})) = \frac{(1+1)(0.5 \cdot 0.5 + 0.4 \cdot 0.4 + 0.01 \cdot 0.01 + \dots + 0.01 \cdot 0.01)}{1+1} = 0.411$.

We can see that expected F_1 prefers $A_{1,2}$ answer in this case. Answer $A_{1,2,\dots,12}$ gets a lower score than $A_{1,2}$, as its combined weight is 1, but it associates lower confidence with the most likely correct elements v_1 and v_2 . \square

3.3 Maximizing Expectation

Let us compute the optimal combination of weights $W = \{w_1, w_2, \dots, w_n\}$ that the algorithm should choose in order to maximize the expected value of F_α defined by Eq. (10). First, we will need to prove an auxiliary lemma. We note that there are multiple different ways to prove the subsequent lemmas and theorems, and we have chosen proofs that we thought would appeal to most of the readers.

Lemma 1. Let a be an attribute with $V = \{v_1, v_2, \dots, v_n\}$ and $P = \{p_1, p_2, \dots, p_n\}$, where $p_1 \geq p_2 \geq \dots \geq p_n > 0$. Then, there exists an optimal answer $W = \{w_1, w_2, \dots, w_n\}$ that maximizes F_α such that $1 \geq w_1 \geq w_2 \geq \dots \geq w_n \geq 0$.

Proof. To prove it, it is sufficient to show that if, for any answer W , there exist $w_i, w_j \in W$, where $i < j$, such that $w_i < w_j$ then by swapping the values of w_i and w_j the expected F_α can only increase or remain unchanged.

Since $w_i < w_j$ thus w_j can be represented as $w_j = w_i + \Delta_w$, where $\Delta_w > 0$ is some real positive value. Since $p_i \geq p_j$ thus p_i can be represented as $p_i = p_j + \Delta_p$, where $\Delta_p \geq 0$ is some nonnegative real value. Observe that exchanging values of w_i and w_j will not change the denominator of $\alpha + \sum_{i=1}^n w_i$ in Eq. (10). However, the numerator $(1 + \alpha) \sum_{i=1}^n p_i w_i$ will increase. Specifically, the part $p_i w_i + p_j w_j$ will now become $p_j w_i + p_i w_j$. Thus, before it used to be $p_i w_i + p_j w_j = (p_j + \Delta_p)w_i + p_j(w_i + \Delta_w) = p_j w_i + \Delta_p w_i + p_j w_i + p_j \Delta_w$. Now it becomes $p_j w_i + p_i w_j = p_j w_i + (p_j + \Delta_p)(w_i + \Delta_w) = p_j w_i + p_j w_i + p_j \Delta_w + \Delta_p w_i + \Delta_p \Delta_w$. Therefore it has increased by $\Delta_p \Delta_w \geq 0$. Hence such a swap will either increase $\mathbb{E}(F_\alpha(W, a))$ or leave it unchanged. \square

The next theorem explains the Maximization of Expectation (MoE) strategy⁷ for computing an optimal answer that maximizes $\mathbb{E}(F_\alpha(W, a))$.

Theorem 1. For an attribute a_j with $V = \{v_1, v_2, \dots, v_n\}$ and $P = \{p_1, p_2, \dots, p_n\}$, where $p_1 \geq p_2 \geq \dots \geq p_n > 0$, the expected F_α will be maximized by choosing weights W such that $w_i = 1$ for $i = 1, 2, \dots, k$ and $w_i = 0$ for $i = k + 1, k + 2, \dots, n$ and where $k = \operatorname{argmax}_k \frac{(1+\alpha) \sum_{i=1}^k p_i}{\alpha+k}$.

Proof. This theorem states that there is an optimal answer that will consist of k values of 1 trailed by $n - k$ values of 0:

$$W = \{\underbrace{1, 1, \dots, 1}_k, \underbrace{0, 0, \dots, 0}_{n-k}\}. \quad (11)$$

To prove it observe that from Lemma 1 we know there exists an optimal answer W such that $1 \geq w_1 \geq w_2 \geq \dots \geq w_n \geq 0$. If this answer is $w_1 = w_2 = \dots = w_n = 1$ then it does conform to (11). If there exist at least one value among w_1, w_2, \dots, w_n that is not equal to 1, then we can always represent W as:

$$W = \{\underbrace{1, 1, \dots, 1}_{j-1}, w_j, w_{j+1}, \dots, w_n\}. \quad (12)$$

⁷We have chosen MoE as the name for the strategy, and not ME or EM, in order to avoid potential confusion, since ME and EM are acronyms for other well-established algorithms.

where w_j is the first value that is not equal to 1. If $w_j = 0$, then from Lemma 1 it follows that $w_j = w_{j+1} = \dots = w_n = 0$ and therefore (11) holds for that case as well.

Let us now consider the remaining case where $0 < w_j < 1$ and where replacing w_j with 0 (or 1) will lead to a non-optimal answer. We will prove by contradiction that such a case cannot exist.

Let F_j be the expected F value for weights W . Let F_0 be the expected F value for the case where w_j is replaced by 0 and F_1 where w_j is replaced by 1. Then, given that F_j is the optimal value of F , whereas F_0 is not, it should follow that:

$$F_j - F_0 > 0. \quad (13)$$

We will now show that from $F_j > F_0$ it will follow that $F_1 > F_j$, which contradicts that F_j is the optimal value. To demonstrate that, let us compute which conditions must hold to satisfy (13). If we rename $\sum_{i=1}^n p_i w_i$ as a and $\alpha + \sum_{i=1}^n w_i$ as b , then from Eq. (10) we have:

$$F_j = (1 + \alpha) \frac{a}{b}, \quad (14)$$

$$F_0 = (1 + \alpha) \frac{a - p_j w_j}{b - w_j}, \quad (15)$$

$$F_1 = (1 + \alpha) \frac{a + p_j(1 - w_j)}{b + (1 - w_j)}. \quad (16)$$

Then

$$\begin{aligned} F_j - F_0 &= (1 + \alpha) \frac{a(b - w_j) - (a - p_j w_j)b}{b(b - w_j)} \\ &= (1 + \alpha) \frac{w_j(p_j b - a)}{b(b - w_j)}. \end{aligned} \quad (17)$$

Thus, for (13) to hold it must follow that:

$$p_j b - a > 0. \quad (18)$$

But, on the other hand, then

$$\begin{aligned} F_1 - F_j &= (1 + \alpha) \frac{ab + bp_j(1 - w_j) - ab - a(1 - w_j)}{(b + (1 - w_j))b} \\ &= (1 + \alpha) \frac{bp_j - bp_j w_j - a + aw_j}{(b + (1 - w_j))b} \\ &= (1 + \alpha) \frac{(bp_j - a)(1 - w_j)}{(b + (1 - w_j))b} \\ &> 0. \end{aligned} \quad (19)$$

That is, $F_1 > F_j$ and thus F_j cannot be an optimal value, which contradicts that W was an optimal answer.

From equation Eq. (10) we can compute the expected F value for answer (11) as $\mathbb{E}(F_\alpha(W, a)) = \frac{(1+\alpha) \sum_{i=1}^k p_i}{\alpha+k}$. While the above discussion does not specify how to compute the optimal value of k , it can be determined by choosing $k = \operatorname{argmax}_k \frac{(1+\alpha) \sum_{i=1}^k p_i}{\alpha+k}$. \square

It is easy to see that the above solution has linear time complexity of $O(n)$.

Example 3.3.1. Consider the setup of Example 3.1.1. If we set $\alpha = 1$ then the values of $\frac{(1+\alpha)\sum_{i=1}^k p_i}{\alpha+k}$ for $k = 1, 2, \dots, 12$ are: (1) 0.5, (2) 0.6, (3) 0.455, (4) 0.368, (5) 0.31, (6) ≈ 0.27 , (7) 0.2375, (8) 0.21(3), (9) 0.194, (10) 0.17(81), (11) 0.165, and (12) ≈ 0.15 . Since 0.6 is the highest value, the algorithm will pick $A_{1,2} = \{v_1, v_2\}$ as its answer. \square

Example 3.3.2. We now know that **Attr-MoE** will pick $A_{1,2} = \{v_1, v_2\}$ as its answer for the setup of Example 3.1.1 where the ground truth was $g = v_2$. Let us see which answers will be chosen by different baseline algorithms from Section 2.5. **Attr-Top-1** will pick $A = \{v_1\}$. Without knowing the actual ground truth, this answer is not a bad pick since v_1 has the 50% chance of being the correct answer – and in fact it was a close second choice according to expected F_1 . But since $g = v_2$ the answer does not include the correct element. **Attr-All** will pick all 12 elements $A = \{v_1, v_2, \dots, v_{12}\}$ only one of which, v_2 , is the correct answer. **Attr-Thr- τ** depends on the value of τ , e.g., for $\tau = 0.45$ it will return the same answer as **Attr-Top-1**, for $\tau = 0.10$ it will return answer as **Attr-MoE**, for $\tau = 0$ it will return the same answer as **Attr-All**. The situation for **Attr-Top-K** is similar. \square

We should realize that while in the above *instance* of the problem **Attr-MoE** performs well, there could be a different instance where some basic technique will outperform **Attr-MoE**. For example, if the ground truth were $g = v_1$ instead of $g = v_2$, then **Attr-Top-1** would perform better on that particular instance, as its answer $A = \{v_1\}$ would match the ground truth exactly. However, when the quality is measured as the average over many queries, **Attr-MoE** will outperform the baseline techniques since it is optimal in the expected sense. We can also note that the proposed **Attr-MoE** approach is essentially equivalent to **Attr-Thr- τ** and **Attr-Top-K** methods described in Section 2.5, but where τ and K are decided *automatically* per uncertain attribute based on the given α and pmf's being observed. However, observe that there is a semantic difference between varying α in F_α and the value of the threshold. Namely, the analyst would typically know beforehand how to set α (e.g., $\alpha = 1$), but not how to set the threshold.

3.4 Efficiency Optimization

The next lemma addresses the efficiency aspect of the problem by allowing to avoid making n computations of $\frac{(1+\alpha)\sum_{i=1}^k p_i}{\alpha+k} = (1+\alpha)G_k$ to determine optimal k .

Lemma 2. Once function $G_k = \frac{\sum_{i=1}^k p_i}{\alpha+k}$, where $p_1 \geq p_2 \geq \dots \geq p_n > 0$, starts decreasing for some k , it continues to monotonically decrease for $k+1, k+2, \dots, n$.

Proof. Let us rename $\sum_{i=1}^k p_i$ as a and $\alpha+k$ as b . Then $G_k = \frac{a}{b}$, $G_{k+1} = \frac{a+p_{k+1}}{b+1}$, and $G_{k+2} = \frac{a+p_{k+1}+p_{k+2}}{b+2}$. We will show that from $G_{k+1} < G_k$ it follows $G_{k+2} < G_{k+1}$. Observe that:

$$(G_{k+1} < G_k) \iff (p_{k+1}b < a) \tag{20}$$

$$(G_{k+2} < G_{k+1}) \iff (p_{k+2}(b+1) < p_{k+1} + a). \tag{21}$$

From Ineq. (20) it follows that $p_{k+1}(b+1) < p_{k+1} + a$. Given that, $p_{k+2} \leq p_{k+1}$, it holds that $p_{k+2}(b+1) \leq p_{k+1}(b+1) < p_{k+1} + a$. Thus from (20) follows (21). \square

From Lemma 2 it follows that to find the optimal value of G_k it is sufficient to find k for which G_k starts to decrease and if there is no such k it means G_n is the optimal value. This will allow to find the $\mathbb{E}(F_\alpha(W, a))$, since it is equal to $(1 + \alpha)G_k$ at optimal k . Given that G_k can be computed very effectively incrementally from G_{k-1} , this means for smaller values of n the search for k can stop as soon as the first decrease happens.

Example 3.4.1. Consider Example 3.3.1. We can observe that the value of $\frac{(1+\alpha)\sum_{i=1}^k p_i}{\alpha+k}$ drops from 0.6 to 0.455 as k changes from 2 to 3 and then continues to monotonically decrease. Hence, the algorithm does not need to scan all 12 items, it can stop after first 3 steps. \square

4. OBJECT SELECTION QUERY

Given an object selection query Q the goal is for each object $O_i \in \mathcal{O}$ to decide whether it should be chosen for Q and put in the answer set A , such that the result quality $F_\alpha(A, Q)$, defined by Eq. (3), is maximized. Let G_Q be the ground truth answer for query Q , where G_Q is not known to the algorithm beforehand. Then the goal of the algorithm can be viewed as that of finding answer A that is as close to G_Q as possible.

4.1 Query Unaware Strategy: Attribute-MoE

The object selection task can be accomplished by Attribute-MoE algorithm. It would first choose attribute values for each object $O_i \in \mathcal{O}$ as described in Section 3, with the same value of α for F_α . Object O_i will be included in the answer set if it satisfies the query Q based on the chosen values for the attributes. While Attribute-MoE strategy optimizes the quality of attribute value selection (as opposed to the desired quality of object selection) nevertheless intuitively it should lead to good results. Thus, it can be used as a baseline strategy to compare with.

Attribute-MoE strategy is a query-unaware strategy as it does not take into account the actual query Q and the relevant frequencies of various attribute values v_{ij} for all i and j across the objects. Notice, for instance, that if an attribute value v_{ij} is infrequent among objects in \mathcal{O} then for query $Q = v_{ij}$, whose goal is to get all objects that have an attribute with value v_{ij} , the cardinality of A and G in Eqs. (1), (2), and (3) will be small. Therefore, given the interactions of A and G in these formulas, an incorrect assignment of an infrequent attribute will lead to much greater impact on F_α compared to that of a frequent attribute. Hence, we will develop a query-driven strategy, which we will refer to as Maximization of Expectation (MoE) strategy for object selection.

4.2 Object Probability

The algorithms for object selection that we will discuss in the subsequent sections are based on computing for each object $O_i \in \mathcal{O}_Q$ its probability $p_i = \mathbb{P}(O_i \in G_Q | Q, \mathcal{D})$ to belong to the ground truth answer G_Q for query Q , given query Q and dataset \mathcal{D} . For notational convenience we will refer to $\mathbb{P}(O_i \in G_Q | Q, \mathcal{D})$ simply as $\mathbb{P}(O_i \in G_Q)$. In this section we consider how this probability can be computed under the assumption that there is no dependence among attributes of an object. In Section 4.8 we will demonstrate how the knowledge of dependencies in data,

when available, can help us to further improve the quality of the result set.

As an example of computing $p_i = \mathbb{P}(O_i \in G_Q)$, let us start by considering single-term queries. The task of a single-term query $Q = q$ is to get all objects whose attributes include q as a value. For instance, for an image database this would mean selecting all the images whose tags include q . For this query, probability $\mathbb{P}(O_i \in G_Q)$ is equivalent to the probability that at least one of the attributes of O_i has value q . If object O_i has only one attribute a then $\mathbb{P}(O_i \in G_Q) = \mathbb{P}(a = q)$. If O_i has multiple attributes a_1, a_2, \dots, a_k that contain q as one of their possibilities, then

$$\begin{aligned} \mathbb{P}(O_i \in G_Q) &= 1 - \mathbb{P}(O_i \notin G_Q) \\ &= 1 - \mathbb{P}(a_1 \neq q, a_2 \neq q, \dots, a_k \neq q) \\ &= 1 - \prod_{j=1}^k (1 - \mathbb{P}(a_j = q)). \end{aligned} \tag{22}$$

For instance, for the example illustrated in Figure 1 and query $Q = \text{sun}$, we have $\mathbb{P}(O_i \in G_Q) = 1 - (1 - 0.6) \times (1 - 0) \times (1 - 0) = 0.6$.

By and large, methods for computing probability $\mathbb{P}(O_i \in G_Q)$ have been extensively studied in the literature, especially for Boolean queries [Sarma et al. 2008]. In Section 4.7 we will explain how this probability can be computed efficiently for conjunctive and disjunctive queries in the context of the probabilistic representation being used.

To explain the subsequent solutions, it is useful to separate objects in \mathcal{O} into three non-overlapping categories with respect to p_i :

- (1) *Decided as Negative*. This is a set of objects guaranteed not to be in G_Q . That is, $\mathcal{O}_0 = \{O_i \in \mathcal{O} : p_i = 0\}$ and thus for each $O_i \in \mathcal{O}_0$ it follows $O_i \notin G_Q$.
- (2) *Decided as Positive*. This is a set of objects guaranteed to be in G_Q . That is, $\mathcal{O}_1 = \{O_i \in \mathcal{O} : p_i = 1\}$ and thus for each $O_i \in \mathcal{O}_1$ it follows $O_i \in G_Q$.
- (3) *Undecided*. This is the remaining category that contain objects that might or might not belong to G_Q . That is, $\mathcal{O}_p = \{O_i \in \mathcal{O} : 0 < p_i < 1\}$.

Naturally, it holds that $\mathcal{O} = \mathcal{O}_0 \cup \mathcal{O}_1 \cup \mathcal{O}_p$, $\mathcal{O}_0 \cap \mathcal{O}_1 = \emptyset$, $\mathcal{O}_0 \cap \mathcal{O}_p = \emptyset$, and $\mathcal{O}_1 \cap \mathcal{O}_p = \emptyset$. Thus G_Q , and consequently each feasible answer A , consist of elements from \mathcal{O}_1 and a subset of elements from \mathcal{O}_p . Thus we define the candidate object set for query Q as $\mathcal{O}_Q = \mathcal{O}_1 \cup \mathcal{O}_p$. Let us sort objects in $\mathcal{O}_Q = \mathcal{O}_1 \cup \mathcal{O}_p$ based on their p_i and then rename all objects in \mathcal{O}_Q as O_1, O_2, \dots, O_n such that $p_1 \geq p_2 \geq \dots \geq p_n > 0$ and where $n = |\mathcal{O}_Q|$.

4.3 Optimal Solution

We now will discuss two solutions to the problem of object selection that are optimal with respect to the expected quality. One is the exponential enumeration and the other one is a polynomial algorithm.

4.3.1 Exponential Enumeration. The overall strategy of the exponential optimal solution is to enumerate over each feasible answer A and find one that maximizes the expected F measure for the given query Q . That is, the goal is to find

$$A_{opt} = \operatorname{argmax}_A \mathbb{E}(F_\alpha(A, Q)). \tag{23}$$

Here, each answer A is a subset of \mathcal{O}_Q and A_{opt} is the optimal answer in the expected sense. To compute the expected F_α measure the algorithm will enumerate over all potential ground truth answers G_Q , which are subsets of \mathcal{O}_Q . For each G_Q it will compute F-measure value under assumption that it is the ground truth. The expected F-measure will be the probabilistic average of these values. Naturally, this brute-force solution has an exponential computational cost and thus is infeasible in practice. Hence, the question arises whether a polynomial algorithm exists for computing A_{opt} .

4.3.2 Special Case of Empty-Set Answer. Let us consider computing $\mathbb{E}(F_\alpha(A, Q))$ for the case where the answer contains no elements, that is, $A = \{\}$ and $k = 0$ where $k = |A|$. The formulas for this special case will be different from that of the generic case where $k \geq 1$. This will allow us to focus only on the generic case in the subsequent discussion. When $k = 0$, precision of A is always computed as 1, since the answer does not contain incorrect elements. The value of recall of A is computed as 1 only if the cardinality of the ground truth set is zero, $|G| = 0$, since A correctly represent all of the elements from G . That way, when $G = \emptyset$ and the answer is $A = \emptyset$, the answer is correct and thus $F_\alpha(A) = 1$. If, however, $|G| \geq 1$, then recall of A_0 is zero, since it does not return any of the correct elements of G . Given that the probability that $|G| = 0$ can be computed as $\prod_{i=1}^n (1 - p_i)$, for $k = 0$ we have:

$$\mathbb{E}(Precision(A, Q)) = 1, \quad (24)$$

$$\mathbb{E}(Recall(A, Q)) = \prod_{i=1}^n (1 - p_i), \quad (25)$$

$$\mathbb{E}(F_\alpha(A, Q)) = \prod_{i=1}^n (1 - p_i). \quad (26)$$

4.3.3 Polynomial Solution. Now we will show a polynomial time algorithm for computing the optimal answer A_{opt} defined in (23). It will be based on [Li and Deshpande 2009], where the authors solve the problem of finding a consensus world for an and/xor tree probabilistic representation under Jaccard similarity metric.

Similar to [Li and Deshpande 2009], for any feasible answer A , where $|A| \geq 1$, let us define a generating polynomial $\mathcal{P}(x, y)$:

$$\mathcal{P}(x, y) = \prod_{i:O_i \in A} ((1 - p_i) + p_i x) \prod_{i:O_i \notin A} ((1 - p_i) + p_i y). \quad (27)$$

After computing the product, the polynomial will have the form:

$$\mathcal{P}(x, y) = \sum_{ij} c_{ij} x^i y^j, \quad (28)$$

where x and y are variables and c_{ij} are the coefficients. By construction, these coefficients correspond to the combined probability of all the ground truth answers such that their size is $i + j$ and they have exactly i elements from A and j elements not from A . Such cases correspond to precision of $\frac{i}{i+j}$, where $k = |A|$, recall⁸ of $\frac{j}{i+j}$,

⁸If $i+j=0$ recall is always 1, as all elements are always recalled.

and

$$F_\alpha(A, Q) = m_{ij} = \frac{(1 + \alpha)i}{\alpha(i + j) + k}. \quad (29)$$

Consequently, $\mathbb{E}(F_\alpha(A, Q))$ for answer can be computed as:

$$\mathbb{E}(F_\alpha(A, Q)) = \sum_{ij} c_{ij} m_{ij}, \quad (30)$$

or, equivalently

$$\mathbb{E}(F_\alpha(A, Q)) = \sum_{ij} c_{ij} \frac{(1 + \alpha)i}{\alpha(i + j) + k}. \quad (31)$$

We will now prove an auxiliary lemma that will allow us to enumerate only over a set of $n + 1$ possible answers, instead of 2^n , in order to choose A_{opt} .

Lemma 3. For any pair of objects $O_i \in A_{opt}$ and $O_j \notin A_{opt}$ it holds that $p_i \geq p_j$.

Proof. Assume that there exists $A_1 = A_{opt}$ such that $O_i \in A_1$, $O_j \notin A_1$ and it holds that $p_i < p_j$. Let $A_2 = A \cup \{O_j\} \setminus \{O_i\}$. Let $\mathcal{P}_1(x, y)$ and $\mathcal{P}_2(x, y)$ be the generating polynomials for A_1 and A_2 . Then we can use these polynomials as in Eq. (30) to encode $\mathbb{E}(F_\alpha(A_1, Q))$ and $\mathbb{E}(F_\alpha(A_2, Q))$. Their difference $\mathbb{E}(F_\alpha(A_1, Q)) - \mathbb{E}(F_\alpha(A_2, Q))$ will simplify to $\sum_{ij} a_{ij}(m_{i,j+1} - m_{i+1,j})$, where $a_{ij} \geq 0$ are certain coefficients. Given the definition of m_{ij} in Eq. (29), it follows that $m_{i,j+1} - m_{i+1,j} < 0$ for any $i, j \geq 0$ and thus $\mathbb{E}(F_\alpha(A_1, Q)) < \mathbb{E}(F_\alpha(A_2, Q))$ and A_1 cannot be the optimal answer. \square

Recall that \mathcal{O}_Q is the candidate set of objects where objects are sorted based on their probabilities, such that $p_1 \geq p_2 \geq \dots \geq p_n > 0$. We can construct answer A_k of size $|A_k| = k$ by choosing k first objects of \mathcal{O}_Q such that $A_k = \{O_1, O_2, \dots, O_k\}$. From Lemma 3 it follows that A_{opt} is one of these A_k for $k = 0, 1, \dots, n$. Consequently, to determine A_{opt} , the algorithm can iterate over A_k and use (26) and (31) to compute A_{opt} .

Discussion. Let us compute the computational complexity of the above solution. The algorithm iterates over A_k answers for $k = 0, 1, \dots, n$ and for each A_k it computes the generating polynomial. The best algorithm we are aware of for computing the coefficients of this polynomial is based on using FFTs and has $O(n^2 \log n)$ complexity [Moenck 1976]. Therefore, the overall complexity of the algorithm is $O(n^3 \log n)$. Thus, we have proven that the optimal answer can be determined in polynomial time.

However, given that n can be large in practice⁹, neither exponential nor the above polynomial algorithm are feasible when n can be large. There is a potential direction¹⁰ that could reduce the cost of the polynomial solution to $O(n^2 \log^2 n)$,

⁹We have routinely observed $n \geq 10,000$ in our data.

¹⁰It might be possible to prove an analog of Lemma 2 for this case. It would state that once you observe the first case where $\mathbb{E}(F_\alpha(A_k, Q)) > \mathbb{E}(F_\alpha(A_{k+1}, Q))$, it should follow that starting from that k function $\mathbb{E}(F_\alpha(A_k, Q))$ will monotonically decrease with the increase of k . If so, then it would be possible to use a binary search among A_k , instead of a linear scan, changing one n factor into $\log n$.

but even that complexity is still impractical. Hence, we do not pursue solutions that find the *exact* optimum answer any further.

In the next section we propose a new object selection algorithm that has $O(n)$ time complexity while at the same time reaching almost the same result quality as the above two algorithms. The proposed algorithm is based on *estimating* the expected values quickly, instead of spending computational resources to get their exact values.

4.4 Basic-MoE Algorithm

From the previous section we know that the algorithm needs to pick one of the A_k answers, where $k = 0, 1, \dots, n$. Thus, the goal of finding A_{opt} can be viewed as that of finding the right value of k . To design an efficient solution in this section we propose Basic-MoE algorithm. Instead of computing the exact value of $\mathbb{E}(F_\alpha(A_k, Q))$, Basic-MoE estimates it by first using Taylor series expansion to approximate $F_\alpha(A_k, Q)$ and then computing the expected value of the approximation.

Let $V_i \sim \text{Bernoulli}(p_i)$ be a Bernoulli random variable such that:

$$V_i = \begin{cases} 1 & \text{if } O_i \in G; \\ 0 & \text{if } O_i \notin G. \end{cases} \quad (32)$$

Let us define two Bernoulli sums

$$X_k = \sum_{i=1}^k V_i, \quad Y_k = \sum_{i=k+1}^n V_i. \quad (33)$$

Variable X_k corresponds to the number of ground truth elements in the answer A_k , whereas Y_k corresponds to the number of the ground truth elements that are not part of A_k . It holds that:

$$\mu_{X_k} = \mathbb{E}(X_k) = \sum_{i=1}^k p_i, \quad \mu_{Y_k} = \mathbb{E}(Y_k) = \sum_{i=k+1}^n p_i \quad (34)$$

$$\sigma_{X_k}^2 = \sum_{i=1}^k (1 - p_i)p_i, \quad \sigma_{Y_k}^2 = \sum_{i=k+1}^n (1 - p_i)p_i. \quad (35)$$

For specific values of X_k and Y_k , the corresponding precision, recall and F_α become:

$$\text{Precision}(A_k, Q) = \frac{X_k}{k}, \quad (36)$$

$$\text{Recall}(A_k, Q) = \frac{X_k}{X_k + Y_k}, \quad (37)$$

$$F_\alpha(A_k, Q) = (1 + \alpha) \frac{X_k}{\alpha X_k + \alpha Y_k + k}. \quad (38)$$

Therefore,

$$\mathbb{E}(F_\alpha(A_k, Q)) = (1 + \alpha) \mathbb{E} \left(\frac{X_k}{\alpha X_k + \alpha Y_k + k} \right). \quad (39)$$

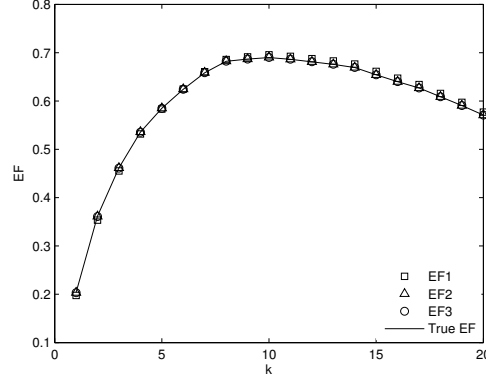


Fig. 3. Comparing $\mathbb{E}(F_\alpha(A_k, Q))$ to $\hat{\mathbb{E}}_i(F_\alpha(A_k, Q))$.

Consider function $f(x, y)$ defined as:

$$f(x, y) = \frac{x}{\alpha x + \alpha y + k}. \quad (40)$$

Its first-order expansion of Taylor series for $f(x, y)$ about the point (a, b) is:

$$f(x, y) \approx f(a, b) + (x - a)f_x(a, b) + (y - b)f_y(a, b). \quad (41)$$

If we set $a = \mu_{X_k}$ and $b = \mu_{Y_k}$ and then take the expectation of both sides, we get:

$$\mathbb{E}(f(x, y)) \approx f(\mu_{X_k}, \mu_{Y_k}). \quad (42)$$

Given that $f(\mu_{X_k}, \mu_{Y_k}) = \frac{\mu_{X_k}}{\alpha(\mu_{X_k} + \mu_{Y_k}) + k}$, from Eq. (39) we have:

$$\begin{aligned} \mathbb{E}(F_\alpha(A_k, Q)) &\approx \hat{\mathbb{E}}(F_\alpha(A_k, Q)) = \\ &= (1 + \alpha) \frac{\mu_{X_k}}{\alpha(\mu_{X_k} + \mu_{Y_k}) + k} = \\ &= (1 + \alpha) \frac{\sum_{i=1}^k p_i}{\alpha \sum_{i=1}^n p_i + k}. \end{aligned} \quad (43)$$

Here, $\hat{\mathbb{E}}(\cdot)$ refers to approximation of $\mathbb{E}(\cdot)$. Similarly, by using the second-order Taylor expansion and then taking the expectation of both sides, we get:

$$\mathbb{E}(f(x, y)) \approx f(\mu_{X_k}, \mu_{Y_k}) + \frac{\sigma_{X_k}^2}{2} f_{xx}(\mu_{X_k}, \mu_{Y_k}) + \frac{\sigma_{Y_k}^2}{2} f_{yy}(\mu_{X_k}, \mu_{Y_k}). \quad (44)$$

From Eq. (44), we can get the corresponding second-order approximation $\hat{\mathbb{E}}_2(F_\alpha(A_k, Q))$ of $\mathbb{E}(F_\alpha(A_k, Q))$. We can repeat the same process for the third order approximation $\hat{\mathbb{E}}_3(F_\alpha(A_k, Q))$.

The best value for k is then chosen as the one that maximizes the approximation of $\mathbb{E}(F_\alpha(A_k, Q))$. The special case of $k = 0$ is solved exactly, by using Eq. (26).

Example 4.4.1. Let $n = 20$, $\alpha = 1$, and $p_i \sim U[0, 1]$. Figure 3 compares $\mathbb{E}(F_\alpha(A_k, Q))$ to $\hat{\mathbb{E}}_i(F_\alpha(A_k, Q))$ for $i = 1, 2, 3$ for various values of k . As we can see all the curves are very close to each other. The maximum deviation from $\mathbb{E}(F_\alpha(A_k, Q))$ is only: $\max_{i,k} |\mathbb{E}(F_\alpha(A_k, Q)) - \hat{\mathbb{E}}_i(F_\alpha(A_k, Q))| = 0.0085$. \square

Computational complexity. We can see that if the hash-sort is utilized to order the probabilities p_i , then the proposed strategy has linear time complexity $O(n)$ for the cases of first- and second-order Taylor approximation. This is since as we go from testing A_k to testing A_{k+1} the new moments $\mu_{X_{k+1}}, \sigma_{X_{k+1}}^2$ etc can be recomputed from the previous ones $\mu_{X_k}, \sigma_{X_k}^2$ incrementally in $O(1)$ time. We also have determined empirically that the results for the first, second, and third order Taylor approximations are very close to each other in practice and that all of them very closely approximate the true $\mathbb{E}(F_\alpha(A_k, Q))$ value. Hence, we focus primarily on the first order approximation given by Eq. (43), since it is the fastest to compute.

4.5 Hard Error Bound on Basic-MoE

It should be noted that while Taylor approximations used by Basic-MoE algorithm tend to find high-quality answers, these answers are not always equal to A_{opt} as explained by the following lemma.

Lemma 4. Answers found by Basic-MoE are not always equal to A_{opt} .

Proof. Assume that $\mathcal{O}_Q = \{O_1, O_2\}$, $p_1 = 0.50$, $p_2 = 0.26$ and that the first-order Taylor expansion is utilized. Suppose that α in F_α is set as $\alpha = 1$. Then, Basic-MoE will choose $A_1 = \{O_1\}$ as its answer. However, the optimal answer is $A_2 = \{O_1, O_2\}$. \square

Nevertheless, in general the first order Taylor expansion leads to highly accurate results, especially for large values of n and consequently large values of $\mu = \mu_{X_k} + \mu_{Y_k}$, as stated by the following theorem.

Theorem 2. For fixed $0 < \alpha < 1$ suppose $k \geq 1$ and $\mu_{X_k} + \mu_{Y_k} \rightarrow \infty$. Then

$$\mathbb{E}(F_\alpha(A_k, Q)) - \hat{\mathbb{E}}_1(F_\alpha(A_k, Q)) \rightarrow 0.$$

In other words, as long as $\mu_{X_k} + \mu_{Y_k}$ is sufficiently large, the approximate expectation of $F_\alpha(A_k, Q)$ will become arbitrarily close to the actual expectation. Recall that for the special case of $k = 0$ the error of Basic-MoE is zero, since it treats that case separately and finds the exact value of $\mathbb{E}(F_\alpha(A_0, Q))$. This theorem considers only $k \geq 1$.

of *Theorem 2*. Simple algebra using Eqs. (39) and (43) yields:

$$\begin{aligned} \frac{|\mathbb{E}(F_\alpha(A_k, Q)) - \hat{\mathbb{E}}_1(F_\alpha(A_k, Q))|}{1 + \alpha} &= |\mathbb{E}(b_1 Z_1 - b_2 Z_2)| \\ &\leq b_1 |\mathbb{E}Z_1| + b_2 |\mathbb{E}Z_2|, \end{aligned}$$

where we denote

$$\begin{aligned} b_1 &= \frac{\alpha \mu_{Y_k} + k}{\alpha \mu + k}, & Z_1 &= \frac{X_k - \mu_{X_k}}{\alpha(X_k + Y_k) + k}, \\ b_2 &= \frac{\alpha \mu_{X_k}}{\alpha \mu + k}, & Z_2 &= \frac{Y_k - \mu_{Y_k}}{\alpha(X_k + Y_k) + k}, \end{aligned}$$

and $\mu = \mu_{X_k} + \mu_{Y_k}$. Note that

$$\mathbb{E}Z_1 = \mathbb{E} \left[Z_1 - \frac{X_k - \mu_{X_k}}{\alpha\mu + k} \right] = \frac{\alpha}{\alpha\mu + k} \mathbb{E}W_1,$$

with

$$W_1 = \frac{(X_k - \mu_{X_k})(\mu - X_k - Y_k)}{\alpha(X_k + Y_k) + k}.$$

For any constant $t > 0$ we have

$$|\mathbb{E}W_1| \leq \mathbb{E}|W_1| = \mathbb{E}(|W_1|1_{X_k+Y_k>t}) + \mathbb{E}(|W_1|1_{X_k+Y_k\leq t}),$$

where $1_{(\cdot)}$ denotes the indicator function. Moreover,

$$\begin{aligned} \mathbb{E}(|W_1|1_{X_k+Y_k>t}) &\leq \frac{\mathbb{E}|(X_k - \mu_{X_k})(\mu - X_k - Y_k)|}{\alpha t + k} \\ &\leq \frac{\sqrt{\mathbb{E}(X_k - \mu_{X_k})^2} \sqrt{\mathbb{E}(\mu - X_k - Y_k)^2}}{\alpha t + k} \\ &\leq \frac{\sqrt{\mu_{X_k} \mu}}{\alpha t + k} \leq \frac{\mu}{\alpha t + k}. \end{aligned}$$

Note that it holds $|W_1| \leq |\mu - X_k - Y_k| \frac{\max(X_k, \mu_{X_k})}{\alpha(X_k + Y_k) + k} \leq \max(\alpha^{-1}, \mu/k) |\mu - X_k - Y_k|$. Hence, if $t < \mu$ then

$$\begin{aligned} \mathbb{E}(|W_1|1_{X_k+Y_k\leq t}) &\leq \max\left(\frac{1}{\alpha}, \frac{\mu}{k}\right) \mu \mathbb{P}(X_k + Y_k \leq t) \\ &\leq \max\left(\frac{1}{\alpha}, \frac{\mu}{k}\right) \mu e^{-(\mu-t)^2/(2\mu)}, \end{aligned}$$

where the second step uses the well-known Chernoff inequality for Bernoulli sums. Choosing $t = \mu/2$ and combining the two bounds, we get

$$\mathbb{E}|W_1| \leq \frac{\mu}{\alpha\mu/2 + k} + \max\left(\frac{1}{\alpha}, \frac{\mu}{k}\right) \mu e^{-\mu/8},$$

which is further bounded by a constant that depends only on α . Hence $\mathbb{E}Z_1 = \frac{\alpha}{\alpha\mu + k} \mathbb{E}W_1 \rightarrow 0$ as $\mu \rightarrow \infty$.

Similarly, letting

$$W_2 = \frac{(Y_k - \mu_{Y_k})(\mu - X_k - Y_k)}{\alpha(X_k + Y_k) + k},$$

we have

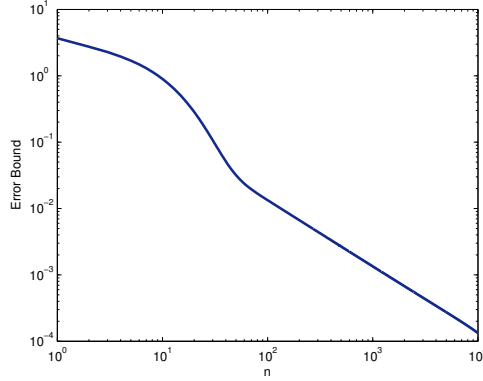
$$\mathbb{E}Z_2 = \frac{\alpha}{\alpha\mu + k} \mathbb{E}W_2.$$

By the same argument, we obtain

$$\mathbb{E}|W_2| \leq \frac{\mu}{\alpha\mu/2 + k} + \max\left(\frac{1}{\alpha}, \frac{\mu}{k}\right) \mu e^{-\mu/8},$$

and hence $\mathbb{E}Z_2 \rightarrow 0$ as $\mu \rightarrow \infty$.

Finally, since $b_1 + b_2 = 1$, we have $b_1|\mathbb{E}Z_1| + b_2|\mathbb{E}Z_2| \rightarrow 0$, as $\mu \rightarrow \infty$, thus proving the claim. \square

Fig. 4. Hard Error Bound on $\hat{\mathbb{E}}_1(F_\alpha(A_k, Q))$.

Remark: the proof implies the following error bound

$$\left| \mathbb{E}(F_\alpha(A_k, Q)) - \hat{\mathbb{E}}_1(F_\alpha(A_k, Q)) \right| \leq \frac{(1 + \alpha)\alpha}{\alpha\mu + k} \left[\frac{\mu}{\alpha\mu/2 + k} + \max\left(\frac{1}{\alpha}, \frac{\mu}{k}\right) \mu e^{-\mu/8} \right].$$

Example 4.5.1. Let $k = \lceil n/2 \rceil$, $\alpha = 1$, and $p_i \sim U[0, 1]$. Figure 4 plots the hard error bound derived by Theorem 2 as a function of n . We can see that as n increases the bound decreases rapidly. Note that this is a *bound* on error and the actual error is less. \square

4.6 Full-MoE Algorithm

Given Lemma 4 and Theorem 2, a higher quality solution than Basic-MoE could be designed by making several observations. Observe that when n is not very large, it is possible to compute the true expected optimal answer A_{opt} , e.g. by using the exact polynomial algorithm. This has allowed us to empirically determine that, in practice, when $n < 12$, A_k found by Basic-MoE is equal to the optimal answer A_{opt} in approximately 90% of the cases. Furthermore, in roughly 99% of the cases, the optimal answer is either A_k , or its direct neighbors A_{k-1} and A_{k+1} . Another important observation is based on measuring the average difference between $F_\alpha(A_{opt})$ and $F_\alpha(A_k)$ as a function of n . We have observed that this difference shrinks rapidly from approximately 14% for the cases where $n < 3$, to less than 1% for $n \geq 6$. Therefore, in terms of quality, Basic-MoE tends to perform similar to the expected optimal solution for the case where n is large, while its performance is somewhat lower for smaller values of n .

Based on these observations, we propose a hybrid approach, to which we refer as MoE. Since enumerations are very fast for small problems sizes, MoE uses the ground-truth enumeration approach discussed in Section 4.3 to choose one among A_k answers for the cases where $n \leq \tau_{enum}$, where we set threshold value $\tau_{enum} = 3$. It employs *smart enumeration* for the cases where $\tau_{enum} < n \leq \tau_{smart}$, where we set $\tau_{smart} = 6$. The smart enumeration algorithm first determines k using Eq. (43) of Basic-MoE. But unlike Basic-MoE, smart enumeration then uses the enumeration algorithm to compute the true (non-estimated) expected F_α values for answers A_{k-1} , A_k , and A_{k+1} . Among the three answers it picks the one that maximizes the

true expected F_α . That way, the smart enumeration is faster than full enumeration, but at the same time it is more accurate than Basic-MoE. Finally, the algorithm utilizes Basic-MoE for the cases where $n > \tau_{smart}$. As the result, the F_α quality of the resulting hybrid algorithm is nearly equal to that of the full enumeration. In terms of computational complexity, since the enumeration algorithm is only applied to cases where n is limited, it remains linear $O(n)$.

4.7 Efficiency Optimizations

Basic-MoE part of the MoE strategy has two natural levels at which its performance can be optimized further. The first (higher) level optimizes the sequential scan among A_k answers. The second (lower) level optimizes the computations of p_i 's – which can be expensive for certain types of queries.

4.7.1 Higher Level Optimization

Lemma 5. Once $\hat{\mathbb{E}}(F_\alpha(A_k, Q))$ starts decreasing for some k it continue to decrease monotonically for $k+1, k+2, \dots, n$. Thus, it is not necessary to scan all of the $k = 1, 2, \dots, n$ values. Instead, one can stop once the first decrease is detected.

Proof. The statement immediately follows from Lemma 2 by observing that that $\frac{\hat{\mathbb{E}}(F_\alpha(A_k, Q))}{1+\alpha}$ has the functional form of $\frac{\sum_{i=1}^k p_i}{\alpha_1+k}$ where $\alpha_1 = \alpha \sum_{i=1}^n p_i$. \square

If a sequential scan is performed, then naturally we do not need to recompute the entire $\sum_{i=1}^k p_i$ for each k when computing $\hat{\mathbb{E}}(F_\alpha(A_k, Q))$. Instead notice that $\frac{\hat{\mathbb{E}}(F_\alpha(A_k, Q))}{(1+\alpha)} = \frac{a_k}{b+k}$ where $a_k = \sum_{i=1}^k p_i$ and $b = \alpha \sum_{i=1}^n p_i$. Thus, to detect the first decrease we simply need to compute b once and compute $a_{k+1} = a_k + p_{k+1}$ incrementally on each iteration. Once the decrease of $\frac{a_k}{b+k}$ is detected, we can get $\hat{\mathbb{E}}(F_\alpha(A_k, Q))$ by multiplying that value of $\frac{a_k}{b+k}$ once by $(1+\alpha)$. Hence, $\hat{\mathbb{E}}(F_\alpha(A_k, Q))$ for a given k can be recomputed efficiently in an incremental fashion from the information stored on the previous $(k-1)$ -th iteration.

4.7.2 Lower Level Optimization. This optimization level deal with optimizing the computation of $p_i = \mathbb{P}(O_i \in G_Q)$. In general such a probability can be computed by the state of the art probabilistic databases for an arbitrary Boolean query and other query types, provided the probabilistic content is initially stored in such a database [Widom 2005; Sarma et al. 2008]. However, for completeness, in this section we present efficient algorithms for computing probability $\mathbb{P}(O_i \in G_Q)$ for conjunctive and disjunctive queries on the specific probabilistic representation being used.

Disjunctive (OR) Queries. The objective of a disjunctive query $Q = q_1 \vee q_2 \vee \dots \vee q_n$ is to get each object $O_i \in \mathcal{O}$ whose attribute values $a_{i1}, a_{i2}, \dots, a_{ik_i}$ include at least one of the values q_i , for $i = 1, 2, \dots, n$. Database indexes could be used very effectively to locate the objects that might qualify. For each such object the probability $\mathbb{P}(O_i \in G_Q)$ is then can be computed as:

$$\mathbb{P}(O_i \in G_Q) = 1 - \prod_{j=1}^{k_i} \left(1 - \sum_{i=1}^n \mathbb{P}(a_j = q_i) \right). \quad (45)$$

```

PROCESS-AND-QUERY-MoE(Q)
1   $\mathcal{O}_Q \leftarrow \text{GET-MATCHES-VIA-INDEX}(Q)$ 
2  for each  $O_i \in \mathcal{O}_Q$  do
3     $\ell \leftarrow \text{GET-PLACEMENT-CASE}(Q, O_i)$ 
4    if  $\ell \neq \text{unknown}$  then
5       $p_i \leftarrow \text{GET-PLACEMENT-PROB}(Q, O_i, \ell)$ 
6    else
7       $p_i \leftarrow \text{GET-POLYNOM-PROB}(Q, O_i)$ 
8    if  $p_i = 0$  then
9       $\mathcal{O}_Q \leftarrow \mathcal{O}_Q \setminus \{O_i\}$ 
10 if  $\mathcal{O}_Q = \emptyset$  then
11   return  $\emptyset$ 

12 HASHSORT-AND-RENAME-OBJECTS( $\mathcal{O}_Q$ )
13  $E_0 \leftarrow \text{GET-EXP-F-ZEROK}(\mathcal{O}_Q)$ 
14  $n \leftarrow |\mathcal{O}_Q|$ 
15 if  $n \leq \tau_{enum}$  then
16    $(A, E) \leftarrow \text{GET-EXP-F-ENUM}(\mathcal{O}_Q)$ 
17 else if  $\tau_{enum} < n \leq \tau_{smart}$  then
18    $(A, E) \leftarrow \text{GET-EXP-F-SMART-ENUM}(\mathcal{O}_Q)$ 
19 else
20    $(A, E) \leftarrow \text{GET-EXP-F-BASIC-MoE}(\mathcal{O}_Q)$ 

21 if  $E_0 > E$  then
22   return  $\emptyset$ 
23 else
24   return  $A$ 

```

Fig. 5. Processing AND Query by MoE.

Computing $\mathbb{P}(O_i \in G_Q)$ as in (45) can be done by a linear scan¹¹ of all possible values of each attribute of O_i . Thus, this type of query could be answered very efficiently.

Conjunctive (AND) Queries. The goal of a conjunctive query $Q = q_1 \wedge q_2 \wedge \dots \wedge q_n$ is to get each object $O_i \in \mathcal{O}$ whose attribute values a_1, a_2, \dots, a_m include all of the q_1, q_2, \dots, q_n . For each object O_i we need to determine the probability $\mathbb{P}(O_i \in G_Q)$ of O_i to belong to the answer set A of Q .

Linear Cost Solution. For conjunctive query $Q = q_1 \wedge q_2 \wedge \dots \wedge q_n$ it often holds that the number of its terms n is limited (bounded by a constant)¹², as large n will result in a very high query selectivity and no objects will be returned as the result of a query.

Theorem 3. For query $Q = q_1 \wedge q_2 \wedge \dots \wedge q_n$, if the number of query terms n is bounded by a constant, then $\mathbb{P}(O_i \in G_Q)$ can be computed in $O(m)$ time, where m is the number of attributes of object O_i .

Proof. To prove that, we will use a generating polynomial, that will be constructed

¹¹Observe that since the number of alternatives per each attribute is typically small, replacing this scan of a tiny array with lookup in a hash table, or similar types of further optimizations, does not lead to a significant improvement.

¹²For instance, currently Google does not allow more than 32 query terms in its search queries.

by multiplying certain terms. A term

$$T_i = (p_{i1}q_1 + p_{i2}q_2 + \cdots + p_{in}q_n + p_{i(n+1)})$$

is constructed per each attribute a_i . Here, q_i 's are variables and p_{ij} 's are coefficients. Coefficients are defined as $p_{ij} = \mathbb{P}(q_j = g_i)$ for $i = 1, 2, \dots, n$ and $p_{i(n+1)} = 1 - \sum_j p_{ij}$, which could be trivially determined from the possible values of a_i and their corresponding probabilities. When multiplying T_i 's, we will be interested in terms in the form of $c \cdot q_1^{i_1} q_2^{i_2} \cdots q_n^{i_n}$. Here c denotes the coefficient and $i_j \in \{0, 1\}$ encodes whether q_j is present or not. For instance, $0.5q_1^0 q_2^1 q_3^1$ encodes $0.5q_2 q_3$. When multiplying $c_1 \cdot q_1^{i_1} q_2^{i_2} \cdots q_n^{i_n}$ by $c_2 \cdot q_1^{j_1} q_2^{j_2} \cdots q_n^{j_n}$ the result is defined as:

$$c_1 \cdot q_1^{i_1} q_2^{i_2} \cdots q_n^{i_n} \times c_2 \cdot q_1^{j_1} q_2^{j_2} \cdots q_n^{j_n} = c_1 \cdot c_2 \cdot q_1^{i_1 \vee j_1} \cdots q_n^{i_n \vee j_n}.$$

For instance, $0.5q_1 q_3 \times 0.2q_1 q_2 = 0.1q_1 q_2 q_3$. Then by multiplying all terms T_i we will get a polynomial:

$$\mathcal{P} = T_1 T_2 \cdots T_m = \sum_{i_1, i_2, \dots, i_n} c_{i_1, i_2, \dots, i_n} q_1^{i_1} q_2^{i_2} \cdots q_n^{i_n}. \quad (46)$$

It is easy to see that the coefficient $c_{1,1,\dots,1}$ that corresponds to $q_1 q_2 \cdots q_n$ will contain the desired probability that the object contains all of $q_1 \wedge q_2 \wedge \cdots \wedge q_n$ terms. Observe that we can compute the generating polynomial by multiplying terms T_i sequentially one by one. At each step k we will get a polynomial \mathcal{P}_k in the form of

$$\mathcal{P}_k = T_1 T_2 \cdots T_k = \sum_{i_1, i_2, \dots, i_n} c_{i_1, i_2, \dots, i_n}^{(k)} q_1^{i_1} q_2^{i_2} \cdots q_n^{i_n}$$

that needs to be multiplied by the next term T_{k+1} . Observe that the number of terms of \mathcal{P}_k cannot exceed 2^n and it does not depend on m . The number of terms in T_{k+1} cannot exceed $n + 1$. This means at each step k , when computing \mathcal{P}_{k+1} as $\mathcal{P}_k \times T_{k+1}$ the algorithm will do a limited number of operations that does not depend on m . Given that the overall number of steps is $m - 1$, the complexity of the algorithm is $O(m)$. \square

Practical Solution for AND Queries. Interestingly, in practice this seemingly challenging query can be often answered very efficiently, often without using the generating polynomial, as explained next. The pseudo-code for the algorithm is illustrated in Figure 5.

Indexing. First, indexing techniques again are used to select the set of qualifying objects \mathcal{O}_Q that contain all of the queried terms. Conjunctive queries are naturally very selective and fewer objects will qualify with the increase of the number of query terms n .

Placement Case-Based Optimizations. Let us consider various *placement cases* that arise with respect to the placement of query terms q_1, q_2, \dots, q_n in object attributes a_1, a_2, \dots, a_m . It turns out that the most frequent placement case that holds for the majority of objects in \mathcal{O}_Q is when:

- Each query term q_j is present exactly once in O_i , and
- Each attribute a_k contain no more than one query term.

Let a_{k_j} be the attribute of O_i wherein q_j is present where $j = 1, 2, \dots, n$. Then for this most frequent case $\mathbb{P}(O_i \in G_Q)$ is simply $\mathbb{P}(O_i \in G_Q) = \prod_{j=1}^n \mathbb{P}(a_{k_j} = q_j)$.

Furthermore, for the majority of the remaining placement cases, it holds that while certain query terms can be present more than once in O_i , no single attribute a_k of O_i contains two or more distinct query terms. That is, each a_k of O_i contains at most one query term. In these cases the exact formula for $\mathbb{P}(O_i \in G_Q)$ can be derived in a straightforward manner.

In the cases that do not belong to the above categories, some of the attributes of O_i will contain at least two distinct query terms. Even then significant optimizations are possible. First, we can identify the top- K (e.g., $K = 10$) most frequent remaining placement cases with respect to how query terms q_1, q_2, \dots, q_m are placed inside attributes a_1, a_2, a_n . We then can *precompute* the right $\mathbb{P}(O_i \in G_Q)$ formulas for them, so when these cases are discovered later on, the right formulas are immediately applied. The remaining (very rare) cases can be handled by the above $O(m)$ solution.

In the next section we empirically evaluate the approaches proposed in this paper and show their significant advantage over the currently used techniques.

4.8 Handling Dependencies in Data

In this section we demonstrate an example of how the proposed object retrieval framework, when provided with additional knowledge of the dependencies in data, can leverage this knowledge to further improve the quality of the result set.

Recall that the proposed object selection algorithm can be viewed as a two-stage process. The first stage determines for each object O_i its probability $p_i = \mathbb{P}(O_i \in G_Q | Q, \mathcal{D})$ to belong to the ground truth answer set G_Q of query Q . The second stage uses these probabilities to compute the final answer A_k . Given the knowledge of certain dependencies, it is frequently possible to improve the first stage of the algorithm by computing probability p_i more accurately, thus resulting in higher quality answers. The way to do that depends on the nature of the captured dependencies and the application domain. In this section we will show how to accomplish that for the speech image tagging scenario described in Section 2.1 by using a supervised learning framework. Similar to [Zadrozny and Elkan 2001; 2002; Niculescu-Mizil and Caruana 2005; Zhang and Yang 2004], we will train a classifier to generate the desired probabilities. Given a feature vector f_i for object O_i and query Q , the classifier will predict whether O_i belongs to G_Q . Many modern classifiers, in addition to their binary yes/no (“+”/“−”) decision of whether object O_i is in G_Q , also output the probability that their decision is correct, which can be transformed into the desired probability $p_i = \mathbb{P}(O_i \in G_Q | Q, \mathcal{D})$. The question thus becomes how to choose features f_i for O_i with respect to Q .

2-term queries. To understand the general idea of the approach, let us first consider an example of a 2-term conjunctive query $Q = q_1 \wedge q_2$. Assume that object O_i is a potential candidate for this query because its attribute a_1 contains q_1 as one of the possible values $v_{11}, v_{12}, \dots, v_{1n}$, and its attribute a_2 contains q_2 among $v_{21}, v_{22}, \dots, v_{2n}$. Suppose that $v_{1k} = q_1$, $v_{2\ell} = q_2$, and the ground truth values for a_1 and a_2 are g_1 and g_2 . Then under the assumption of tag independence p_i will be computed as $p_i = \mathbb{P}(O_i \in G_Q | Q, \mathcal{D}) = p_{1k} p_{2\ell}$, where p_{1k} and $p_{2\ell}$ are the probabilities associated with v_{1k} and $v_{2\ell}$ respectively.

However, in an image dataset the knowledge of correlations among tags can help to compute $\mathbb{P}(O_i \in G_Q|Q, \mathcal{D})$ more accurately. Let \mathcal{D}_{train} be the past corpus of images where the value of all tags are known exactly and there are no uncertain attributes. Let $c(q_1)$ be the number of images in \mathcal{D}_{train} annotated with tag q_1 and $c(q_1, q_2)$ be the number of images annotated with both q_1 and q_2 . Let $c_{12} = c(q_1, q_2)$. For each (v', v'') pair, where $v' \in \{v_{11}, v_{12}, \dots, v_{1n}\}$ and $v'' \in \{v_{21}, v_{22}, \dots, v_{2n}\}$, we can also compute $c(v', v'')$ count. Let c_1 be the highest observed $c(v', v'')$ count, c_2 and c_3 be the top-2 and top-3 counts. The intuition is that the higher $c(v', v'')$ is, the higher the chance is that (v', v'') is the ground truth pair (g_1, g_2) . Therefore, $(c_1 - c_{12}, c_2 - c_{12}, c_3 - c_{12})$ features are indicative of how far is (q_1, q_2) pair from the top-3 pairs in terms of co-occurrences and thus how likely this pair is equal to (g_1, g_2) .

Another important factor to consider is that, for given two terms q_1 and q_2 , it is very rare that a speech recognizer (ASR) will suggest both of them as the possible values of N-best lists for the same image, unless q_1 and q_2 are both the ground truth tags for that image. Consequently, when $c_{12} = c(q_1, q_2)$ is large, the candidate set \mathcal{O}_p consist mostly of “+” objects that satisfy query Q and very few “-” objects that do not satisfy it. Thus, c_{12} is a strong feature in determining $\mathbb{P}(O_i \in G_Q|Q, \mathcal{D})$.

The probabilities p_{1k} and $p_{2\ell}$ assigned by the ASR for q_1 and q_2 in O_i , as well as their differences with the top probabilities p_{11} and p_{21} of a_1 and a_2 , can also be taken into account. The resulting feature vector f_i for O_i regarding Q is:

$$f_i = (p_{1k}, p_{2\ell}, p_{11} - p_{1k}, p_{21} - p_{2\ell}, c_{12}, c_1 - c_{12}, c_2 - c_{12}, c_3 - c_{12}).$$

Another generic way to select feature vector is:

$$f_i = (p_q, p_{top} - p_q, c_{12}, c_1 - c_{12}, c_2 - c_{12}, c_3 - c_{12}),$$

where $p_q = p_{1k}p_{2\ell}$ and $p_{top} = p_{11}p_{21}$ correspond to the probabilities of q_1, q_2 pair and the top pair, computed under the independence assumption.

By training a classifier on a validation set \mathcal{D}_{val} while using training set \mathcal{D}_{train} for computing co-occurrence counts, we can apply the classifier on the test data \mathcal{D}_{test} to compute $p_i = \mathbb{P}(O_i \in G_Q|Q, \mathcal{D})$. As we will see in Section 5, the resulting probabilities are much more accurate compared to those computed under the independence assumption, leading to significantly higher quality of answers.

1-term Queries. A similar approach applies to 1-term queries in the form of $Q = q_1$ when O_i contains at least two uncertain attributes. If one attribute contains q_1 , then we consider the attribute right after it, or right before it if the latter does not exist. For instance, if $O_i.a_1$ contains q_1 , we then will consider $O_i.a_2$. Among the possible values $v_{21}, v_{22}, \dots, v_{2n}$ of $O_i.a_2$, we will find $v_{2\ell}$ that maximizes the co-occurrence with q_1 , that is, $v_{2\ell} = \operatorname{argmax}_{\ell} c(q_1, v_{2\ell})$. The idea is that if q_1 is the ground truth for a_1 , then $v_{2\ell}$ is most likely to be the ground truth for a_2 . We then compute $c_{12} = c(q_1, v_{2\ell})$, and like for 2-term queries, we find c_1 , c_2 , and c_3 as the top-1, top-2, and top-3 of $c(v', v'')$, for all pairs $v' \in \{v_{11}, v_{12}, \dots, v_{1n}\}$ and $v'' \in \{v_{21}, v_{22}, \dots, v_{2n}\}$. Using the same intuition, features $(c_1 - c_{12}, c_2 - c_{12}, c_3 - c_{12})$ are indicative of high likely it is that $q_1 = g_1$ and $v_{2\ell} = g_2$ and thus how likely $O_i \in G_Q$ is. By applying similar logic as for 2-term queries, the final feature vector

is chosen as:

$$f_i = (p_{1k}, p_{11} - p_{1k}, c(q_1), c_1 - c_{12}, c_2 - c_{12}, c_3 - c_{12}),$$

where, as before, k is such that $v_{1k} = q_1$.

m-term Queries. Given an m -term query $Q = q_1 \wedge q_2 \wedge \dots \wedge q_m$ a similar approach could be applied by changing raw correlations $c(q_1, q_2, \dots, q_m)$ into *score* $s(q_1, q_2, \dots, q_m)$. This change is needed since for larger values of m it becomes a challenge to compute nontrivial statistics where $c(q_1, q_2, \dots, q_m) \neq 0$. Instead, a scoring function can compute $s(q_1, q_2, \dots, q_m)$ as the sum of pairwise correlations $c(q_i, q_j)$ for all distinct pairs of q_i and q_j , or it can compute it as the probability to observe an image whose annotation include q_1, q_2, \dots, q_m tags. Different ways for efficient computation of these scores have been discussed in [Kalashnikov et al. 2011]. Assume without loss of generality that $q_1 \in a_1, q_2 \in a_2, \dots, q_m \in a_m$. Let us name the probabilities assigned by ASR as $p_{q_i} = \mathbb{P}(q_i = g_i)$. Let $p_q = p_{q_1} p_{q_2} \times \dots \times p_{q_m}$, which is the probability that $O_i \in G_Q$ under the assumption of independence. Let $p_{top} = p_{11} p_{21} \times \dots \times p_{m1}$, which is the probability of the top combination according to the ASR. Then the feature vector f_i is chosen as:

$$f_i = (p_q, p_{top} - p_q, s_{1m}, s_1 - s_{1m}, s_2 - s_{1m}, s_3 - s_{1m}).$$

Here, $s_{1m} = s(q_1, q_2, \dots, q_m)$ plays the same role as c_{12} for 2-term queries, and s_1, s_2, s_3 are the top 3 combinations according to the scoring function $s(\dots)$, which also could be found efficiently using the branch and bound method proposed in [Kalashnikov et al. 2011].

5. EXPERIMENTAL RESULTS

In this section we empirically study our algorithms for the two types of selection queries. We show that the proposed MoE solutions outperform the existing techniques in terms of quality. We also demonstrate that the efficiency of the proposed approaches is comparable to that of the baseline techniques.

5.1 Experimental setup

5.1.1 Datasets. We have used both synthetic and real datasets in our experiments. Synthetic datasets are employed since they allow us to manually control the level of uncertainty in data and study its effect on various techniques.

—**SynData.** This simple synthetic datasets has 10K objects, where the average number of attributes per object is 4.5. The total number of distinct possible attribute values is 1000. We generate 5 different versions of this dataset. A version is generated by choosing the number of alternatives $N = |V|$ each uncertain attribute can have. Here, the 5 versions correspond to N of 2, 4, 8, 16, 32. After choosing N , the probabilities p_1, p_2, \dots, p_N for N options of an uncertain attribute are assigned by first generating N random numbers r_1, r_2, \dots, r_N , where r_i is drawn from uniform distribution in $[0, 1]$. Probability p_i is then computed as $p_i = \frac{r_i}{\sum_{i=1}^N r_i}$. The *interval method* is then used to decide which option i is the ground truth value for this uncertain attribute. The interval method divides $[0, 1]$ interval into N intervals of length p_1, p_2, \dots, p_N . It then generates a random number in $[0, 1]$ and observes into which interval i it falls. The ground truth

is then assigned to the i -th option. This ensures that each option i can be the ground truth with probability of p_i .

- PubData.** PubData is a real dataset derived from two public-domain sources: CiteSeer and HPSearch. It is a publication dataset that describes authors and their publications. It consists of 11.7K publications and 14.6K authors. The objects in it are publications and attributes are references to authors in these publications, which can be ambiguous, e.g. ‘J. Smith’. The references are resolved using the entity resolution method from [Kalashnikov and Mehrotra 2006], described in Section 2.1. It generates the list of alternatives for each attribute as well as their probabilities.
- MovData.** This is also a real dataset. It is generated from the Stanford’s Movie Dataset and contains, among other things, 11.5K movies and 22.1K directors. The objects in it are movies, the attributes are references to directors in these movies, which can be uncertain. The alternatives for attributes and the corresponding probabilities are provided by the entity resolution method [Kalashnikov and Mehrotra 2006] as well.
- ImgData.** This semi-real dataset imitates a large corpus of images annotated with tags using a speech interface, as explained in Section 2.1. It has 304.9K real images with 8.4K distinct tags from a popular image hosting website Flickr. Each image has 13.7 tags on average. Since the dataset is large, it is infeasible to speak all tags and then apply a Speech Recognizer to generate tag alternatives and the corresponding probabilities. Instead, they are generated synthetically by adapting the methodology from [Hernandez and Stolfo 1995; Menestrina et al. 2006]. Specifically, for each tag its alternative is generated using the Soundex-hash tables. Then N probabilities are generated. First, the highest p_1 is drawn from a Poisson distribution with certain mean and standard deviation. This simulates a speech recognizer being more certain about one alternative. The remaining $N - 1$ probabilities are created by generating $N - 1$ random numbers in $(0, 1]$ interval and then normalizing them such that they add up to $1 - p_1$. The ground truth is assigned by using the interval method.

5.1.2 Baseline Techniques. We will refer to the attribute and object selection approaches proposed in Sections 3 and 4 as **Attr-MoE** and **MoE** respectively. We will compare them to **Attr-Top-1**, **Attr-Top-K**, **Attr-All**, **Attr-Thr- τ** , and **Thr- τ** baseline techniques explained in Section 2.5. These baselines cover a wide spectrum of existing techniques in terms of *quality*, as most of the existing approaches are variants of the above methods. Many recent state of the art strategies instead focus on optimizing the *efficiency* (and not the quality) aspect of the problem. For example, there are many techniques for efficient computations for various top-K and threshold scenarios [Re et al. 2007; Cormode et al. 2009; Zhang and Chomicki 2009].

5.2 Setting Threshold and Alpha

For threshold-based solutions, an important question is to how to set the threshold in our unsupervised settings. This is a nontrivial task since the chosen threshold should lead to high quality results in terms of F_α for both, the given dataset and given α . In the plots in this section, **Attr-Thr- τ** is the approach where the threshold

τ is set *manually* by the domain analyst. Specifically, we set the value of τ to 0.1, 0.25, and 0.5, which are the values that lead to very high F_α 's in our tests. However, we will observe that while setting, for instance, $\tau = 0.1$ leads to excellent F_α results for certain combinations of dataset and α , at the same time τ of 0.1 leads to poor results for other combinations. The purpose of including **Attr-Thr- τ** is to demonstrate how threshold-based techniques are expected to behave in practice, in our unsupervised settings.

As we have explained earlier, parameter α in F_α is also set manually by the domain analyst. It depends on the nature of the application as some applications prefer recall over precision and others prefer the reverse. We will study the performance of various techniques across different values of α .

5.2.1 Query Workload. For attribute value selection, we report average F_α when disambiguating all uncertain attributes in the dataset. For single-term object selection, the queries are all of the possible ground truth tags. Other workloads have been studied extensively as well, but they often result in similar plots and thus they are omitted. For multi-term queries, we test on a query mix consisting of 1-, 2- and 3-term queries in equal proportion. Since the conjunctive queries are very selective, picking completely random and unrelated query terms too often results in a query that returns an empty set. Thus, to make queries more meaningful, when selecting terms for the 2- and 3-term queries we pick terms whose co-occurrence count (computed on the past ground-truth data) was greater than zero.

5.2.2 Measurements. We measure and report the quality achieved by various techniques using the **true F_α measure**. The expected F measure is used by the algorithms only internally and it is not reported. For both attribute and object selection, since we compare a large number of techniques, special care should be taken to avoid very dense graphs with many overlapping curves. Hence, we plot the results in the form of bar charts that show the improvement/gain achieved by the MoE strategy over the other techniques in terms of F_α . Let x be the F_α obtained by MoE and y be the F_α of any other method Y . Then the gain of MoE over Y is computed as $(x - y)$.

To check the statistical significance of our results we have used the two tailed paired t-test. We have found that in rare cases, especially when $|x - y| < 0.01$, the improvement was not statistically significant. Therefore, to focus on only statistically significant results, we plot the improvement as zero when it is not statistically significant.

In the experiments we also indicate the average query execution time, which is computed *per query* and reported in ns or ms, where 1 ms = 10^{-3} sec and 1 ns = 10^{-9} sec.

5.3 Attribute Value Selection

Figure 6 demonstrates the quality improvement by the proposed **Attr-MoE** method over the baseline techniques on the 5 different versions of SynData, for $\alpha = 1$. For threshold-based approaches, we have determined that for our real datasets the most competitive values of the threshold were 0.10 and 0.25. Hence, in our experiments we will focus primarily on **Attr-Thr-0.1** and **Attr-Thr-0.25**. Figure 6 shows that improvement over all the methods is bounded by 10%. The smallest improvement

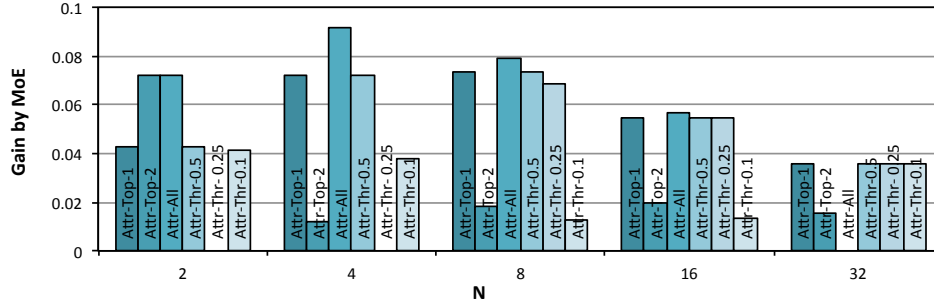


Fig. 6. Attribute Selection on SynData.

is observed over the **Attr-Thr-0.25** method, when the number of options per attribute N is small. As the number of options increases, the improvement over the **Attr-Thr-0.25** becomes visible and reaches nearly 7%. However, for $N = 8$ and $N = 16$, **Attr-Thr-0.1** technique becomes better than **Attr-Thr-0.25**, and similarly the **Attr-Top-2** model is better than the **Attr-Top-1** model when the number of options is large.

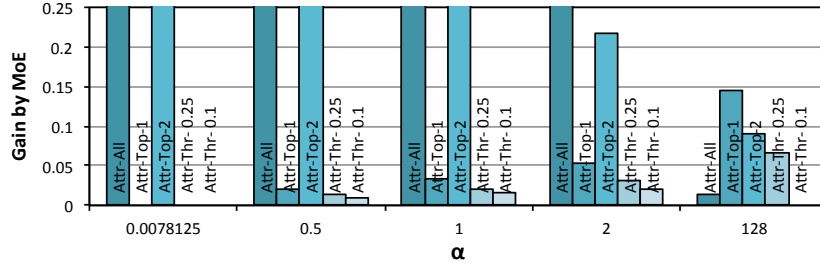
Figures 7 (a), (b), and (c) plot quality improvements achieved by the proposed **Attr-MoE** strategy over baseline methods, on PubData, MovData, and ImgData respectively. The graphs exhibit several important trends, which are explained next.

The first trend is that the proposed MoE-based strategy never performs worse than the baseline techniques, across the board, frequently outperforming baselines by a visible margin. This was also a trend in all of the other experiments. This is as anticipated, since MoE is optimal in the expected sense.

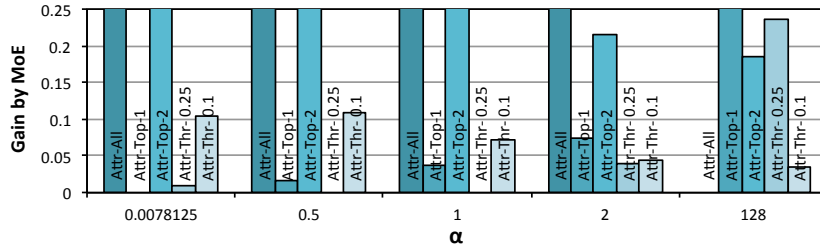
The second trend is that, as α is varied from 2^{-7} to 2^7 , **Attr-All** strategy goes from showing one of the worst results to demonstrating the best results. This is also expected. **Attr-All** always reaches the highest possible recall since it outputs all of the options as its result. Thus, when $\alpha = 2^7$, recall is preferred over precision and **Attr-All** shows excellent results. **Attr-All**, however, reaches one of the worst precisions - since it always includes many wrong elements in the answer. This explains its poor performance for $\alpha = 2^{-7}$ where preference is given to precision.

The third trend is that **Attr-Top-1** tends to behave opposite to **Attr-All**, as it demonstrates excellent results for $\alpha = 2^{-7}$, but poor results for $\alpha = 2^7$. This is because **Attr-Top-1** tend to get the worst recall, since it always chooses only one element. Thus it shows poor results when $\alpha = 2^7$ and recall is preferred over precision. It however show more reasonable results when $\alpha = 2^{-7}$ and thus recall is practically ignored. Table III shows the absolute F_α values reached by **Attr-MoE** on these datasets.

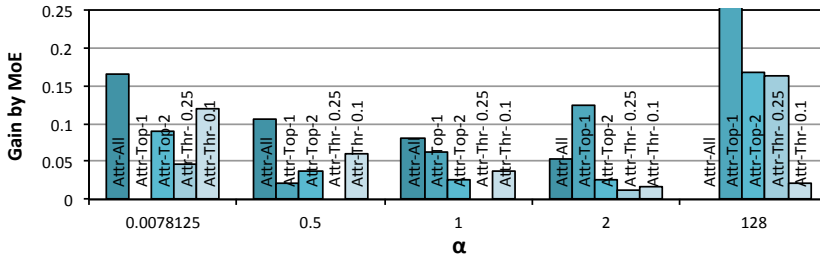
Another important observation from Figures 6 and 7 is that Threshold-based techniques show competitive results. However, no single threshold value works the best across the board. For instance, setting threshold τ to 0.25 would be an excellent choice for MovData when $\alpha = 1$. However, this value does not work as well for some of the other datasets (e.g., PubData) for $\alpha = 1$, where $\tau = 0.10$ is a



(a) PubData



(b) MovData



(c) ImgData

Fig. 7. Attribute Selection. Quality across α .

better choice. The value $\tau = 0.25$ does not also work for the same MovData, but when $\alpha = 2^7$. The same holds for $\tau = 0.10$, as it is an excellent choice for, say, PubData where $\alpha = 1$. Nevertheless, it is not the best technique for many other datasets for $\alpha = 1$ (e.g., ImgData). Thus, setting the right value of the threshold is a challenge.

Notice that the proposed MoE solution is similar to threshold-based strategy, but where the threshold is decided automatically by the algorithm per attribute – based on the given pmf’s of the attribute and α . Thus, MoE gains its advantage by automatically self-adapting to the underlying conditions.

Efficiency. In terms of the efficiency, the proposed Attr-MoE strategy takes 452 ns, 382 ns, and 531 ns per query on average for $\alpha = 2^{-7}$, 1, and 2^7 respectively.

Dataset	$\alpha = 2^{-7}$	$\alpha = 2^{-1}$	$\alpha = 1$	$\alpha = 2$	$\alpha = 2^7$
PubData	0.849	0.865	0.878	0.896	0.989
MovData	0.697	0.715	0.738	0.774	0.975
ImgData	0.536	0.558	0.597	0.660	0.984

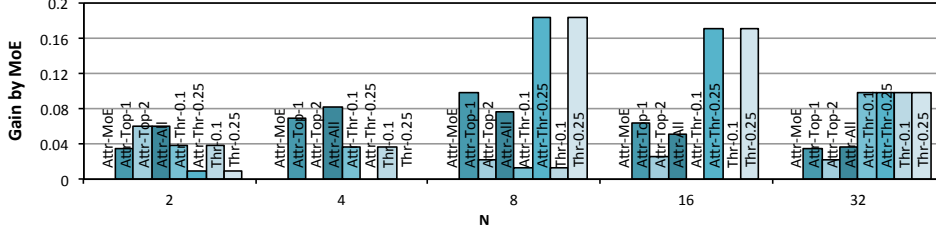
Table III. F_α values reached by **Attr-MoE**.

Fig. 8. Object Selection on SynData.

Attr-Top-1 and **Attr-All** are independent of α in terms of the efficiency and they take 380 ns, 518 ns, and 398 ns on average. Thus, the advantage of **Attr-MoE** in terms of higher quality does not come at the price of much lower efficiency.

5.4 Object Selection Queries

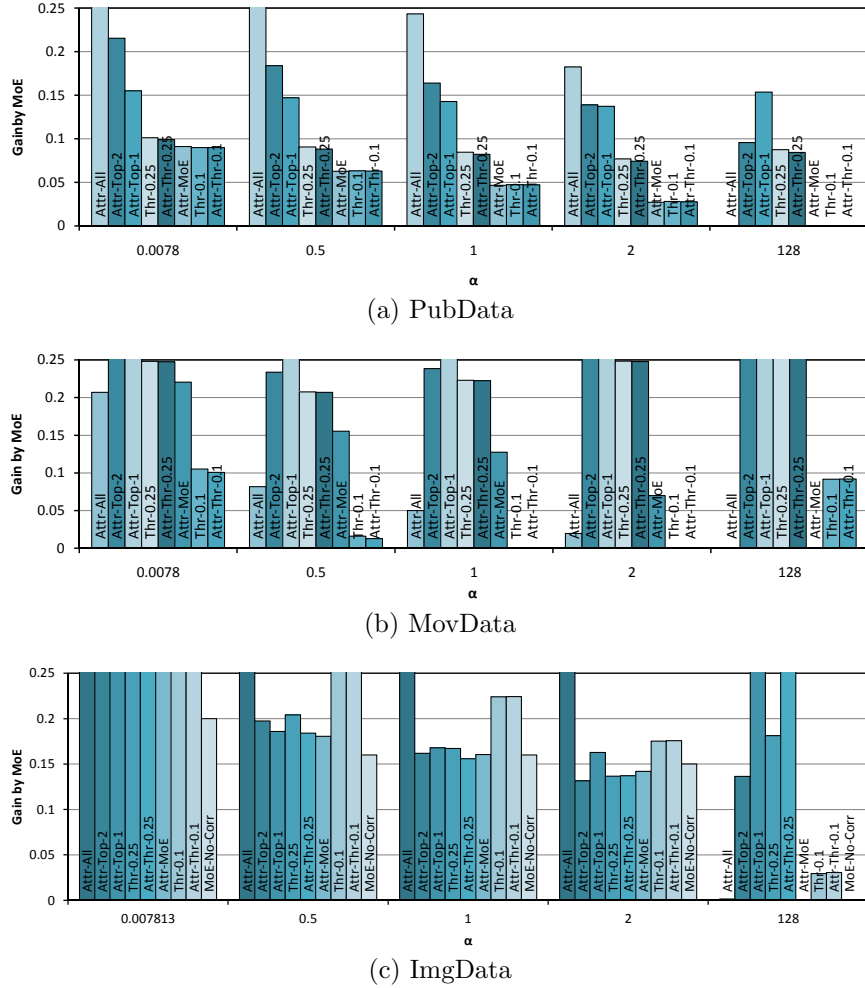
We study single-term queries in Section 5.4.1 and multi-term queries in Section 5.4.2.

5.4.1 Single-Term Queries. Figure 8 shows the quality improvement achieved by MoE approach for object selection on the five different version of the SynData. The proposed **Attr-MoE** and MoE strategies show comparable results and largely dominate other techniques. Both types of thresholding approaches, **Attr-Thr** and **Thr**, demonstrate good results, however no single value of threshold works the best. Instead, $\tau = 0.10$ works the best for smaller values of N , whereas $\tau = 0.25$ is better for larger values of N .

Figures 9 (a), (b), and (c) plot the quality gain of MoE over various techniques as a function of parameter α in F_α for PubData, MovData, and ImgData. They show very similar main trends that are also similar to those of the attribute selection graphs from the previous section, largely for the same reasons. For instance, since MoE strategy tends to get answers that are nearly the same as those of the optimal expected solution, it never gets the results that are worse than the baselines. **Attr-All** tends to show good result for $\alpha = 2^7$, but poor results for $\alpha = 2^{-7}$, and reverse is the case for **Attr-Top-1**. Threshold-based solutions also show competitive results, but no single threshold value works the best, as setting threshold to a good value for some dataset will lead to poor value for another one. This is the case even if the value of α is restricted to, say, $\alpha = 1$ only. Thus it is a challenge to set a good threshold value in unsupervised settings, for any dataset and any α .

One interesting observation is that while **Attr-Top-1** is used very often in practice, in this case it cannot reach reasonable quality compared to the rest of the techniques for a wide range of α !

Figure 9 (d) plots the results of object selection queries for ImgData where dependencies in data are taken into account using the approach described in Section 4.8.

Fig. 9. Object Selection. Quality across α .

MoE-No-Corr corresponds to the MoE approach under the assumption of tag independence. In that figure the performance of MoE-No-Corr tends to be comparable or slightly better than that of the competing strategies. However, by leveraging the knowledge of dependencies, MoE achieves significantly better results compared to all of the competing approaches.

Efficiency. In terms of the execution time, it takes 15.7 ms, 22.7 ms, and 28.7 ms on average per query for the proposed MoE strategy on ImgData for $\alpha = 2^{-7}$, 1, and 2^7 respectively. For Movies dataset these numbers are 5.4 ms, 6.2 ms, and 6.5 ms. On the other hand, it takes 17 ms and 9 ms for Attr-All and Attr-Top-1 strategies and $\alpha = 1$ on ImgData. For Movies dataset these numbers are 2.7 ms, 1.7 ms, and 2.1 ms. As we can see the fastest one is Attr-Top-1, but it also gets much worse results in terms of quality. As expected, due to the required additional

calculations, the performance of MoE is slower compared to the baseline strategies. However, it is quite reasonable if high-quality results are preferred.

5.4.2 Multi-Term Queries. Figures 10 (a)–(d) study the performance of conjunctive and disjunctive multi-term object selection queries on ImgData and PubData. The mixture of 1/2/3-term queries is used for ImgData and a mixture of 1/2-term queries is used for PubData.

For these queries we can see that the relative stand of various strategies is quite similar to that of single-term queries. The proposed MoE strategy outperforms the rest of the techniques while **Attr-Thr-0.1** and **Attr-All** also tend to perform well.

One trend that we observed is that the relative gain of MoE becomes less compared with the single-term case. For instance, since we are using a common workload of relatively popular combination of tags, for the ImgData this translates into answer sets where the number of “+” elements in the candidate sets outnumber the number of “-” elements. Thus, any technique that gives preference to recall, like **Attr-All** or **Attr-Thr-0.1**, will get quite accurate results, whereas techniques that prefer precision, like **Attr-Top-1** will show poor results. We have also considered different synthetic workloads where the “-” elements outweigh “+” elements, and in that case **Attr-Top-1** significantly outperforms **Attr-All** (plots not shown for brevity) but MoE still remained the best strategy.

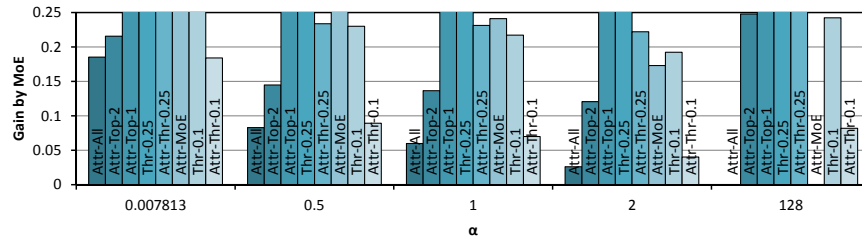
Another observation that we have made is that for certain datasets, e.g. ImgData, the computations under the assumption of attribute independence do not work well and the dependencies should be modeled in data (as described in Section 4.8) to get high quality results, see for example Figures 9(d) and 10(c).

Efficiency. It takes 16.1 ms, 23.1 ms and 25.1 ms on average for MoE for conjunctive queries for $\alpha = 2^{-7}$, 1, and 2^7 respectively, per query. For disjunctive queries these numbers are 29.5 ms, 104.9 ms, and 108.2 ms. On the other hand, it takes 9ms and 5.9ms on average for conjunctive queries and $\alpha = 1$ for **Attr-All** and **Attr-Top-1** respectively. For disjunctive queries these numbers are 96.9 ms and 67.8 ms. As we can see, even though conjunctive queries might require enumeration, they are faster than disjunctive queries since they are much more selective. The fastest one is the **Attr-Top-1** strategy, which gets higher efficiency at the cost of the worst quality results. The relative performance of the proposed MoE strategy is quite reasonable, if high quality results are desired.

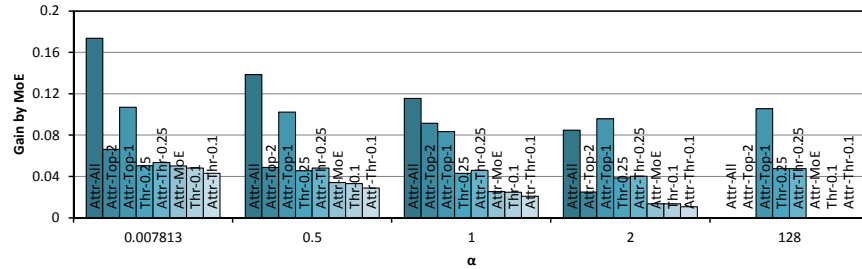
5.5 Miscellaneous Experiments

5.5.1 Stability and Robustness. In this experiment we analyze the stability and robustness of the proposed MoE strategies with respect to the noise in the provided probabilities. This experiment simulates the situation where the underlying data processing techniques do not provide adequate probabilities and no further action is taken to correct or improve these probabilities.¹³ Specifically, let p_1, p_2, \dots, p_n be the probabilities associated with the n options of an attribute. We add noise to these probabilities by first generating a random distribution r_1, r_2, \dots, r_n using the same process as for generating probabilities for the ImgData. We then mix the two

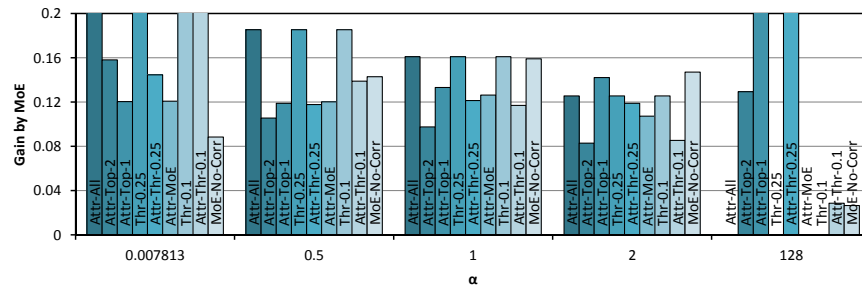
¹³There are many existing supervised learning techniques that allow to adjust probabilities that contain some noise.



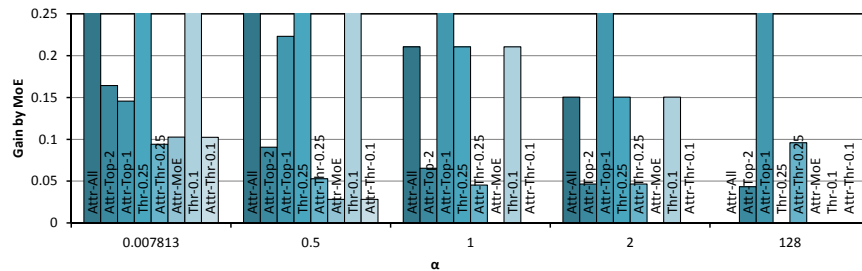
(a) AND-Query on ImgData.



(b) AND-Query on PubData.



(c) OR-Query on ImgData.



(d) OR-Query on PubData.

Fig. 10. Multi-Term Queries.

distributions $a_i = (1 - \gamma)p_i + \gamma r_i$ and then normalize a_i 's such that $\sum_{i=1}^n a_i = 1$. We then use a_i 's instead of p_i 's as the new (noisy) probabilities associated with the n options. That way, as we vary γ from 0 to 1, the original probability distribu-

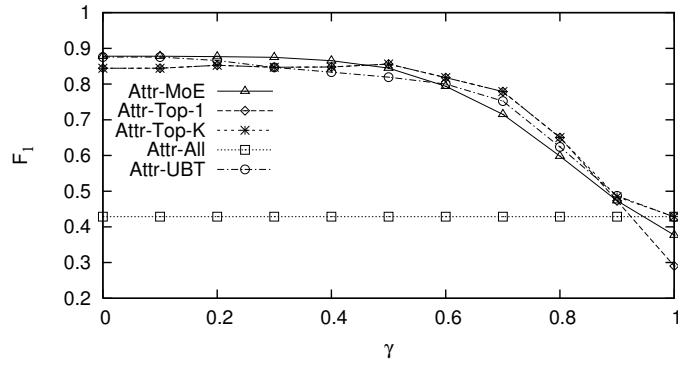


Fig. 11. Noisy Prob.: Attribute Value Selection on PubData.

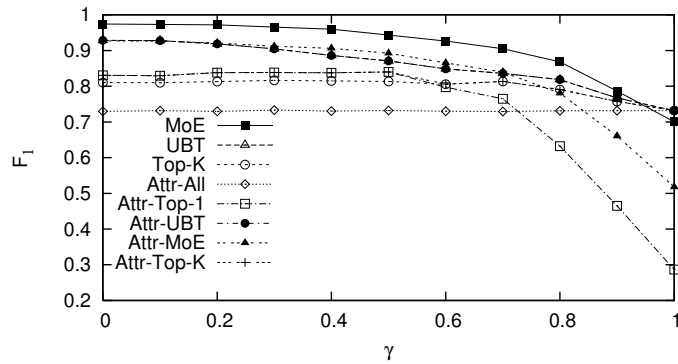


Fig. 12. Noisy Prob.: Object Selection on PubData.

tion gradually transforms into a completely random distribution. This procedure is applied to each uncertain attribute. Figures 11 and 12 plot F_1 -measure for attribute and object selection on Publication dataset as γ is varied in $[0, 1]$ interval. We can see that the results of the proposed MoE technique are quite robust. For attribute value selection, as γ increases the MoE technique initially dominates the other techniques but then becomes slightly worse. Nevertheless, overall it is quite stable as the quality does not drop drastically. The All techniques does not use probabilities (as long as there are no zero probabilities) so its curve is flat. For object selection, MoE largely dominates the other techniques – this has been the case in our other object selection experiments as well. As we can see, a small disturbance in probabilities does not lead to a significant change in relative stand of various methods, but a large amount of noise could do that. Hence, overall MoE strategy is quite robust and stable.

As it can be seen from the figures, in a complete noise environment, most of the methods still have $F_1 = 0.7$ and $F_1 = 0.4$ for object and attribute value selection respectively. Such high quality F_1 values are achieved in a complete noise environment, because for each attribute there are only a few “options” and none of the options are assigned with a zero (or unusually skewed) probabilities by the random

method.

5.5.2 Effect of Optimizations. For attribute value selection, the “stopping early” optimization described by Lemma 2 in Section 3.4 results in 39%, 10%, and 19% improvement of the average query response time for PubData, MovData, and ImgData datasets respectively. For object selection, a similar optimization described by Lemma 5 in Section 4.7.1 results in 5%, 2%, and 9% improvement. When adding the probability computation optimizations from Section 4.7.2, the overall improvements for conjunctive queries are 8%, 4%, and 47%. The improvement is greater for ImgData since, each image can be annotated with the same tag multiple times. This results in more cases that require time-consuming enumerations. Thus, the probability computation optimization helps significantly more by being able to handle these time-consuming cases efficiently.

5.6 Discussion of Results

As expected the proposed MoE-based strategy outperforms the competing techniques under various scenarios. The threshold-based methods were in the second place, achieving competitive result on some of the datasets. Interestingly, widely used Top-K and especially *commonly applied* Top-1-based methods did not perform as well in this context, with Top-1 frequently showing significantly worse results! Both, threshold and Top-K-based techniques require the domain analyst to set the right parameter values, while the proposed MoE approaches do not need any supervision at all. The proposed approach is also quite robust and does not show drastic drop in accuracy if the associated probabilities are noisy. In terms of the execution time, the performance of the proposed MoE techniques is acceptable when compared to other strategies. Thus MoE is a preferred solution when higher quality results are desirable.

6. RELATED WORK

There is very significant amount of related work, especially on probabilistic databases [Widom 2005; Antova et al. 2008; Singh et al. 2008; Dalvi and Suciu 2004; Cheng et al. 2003; Kalashnikov et al. 2006; Ma et al. 2008; Cheng et al. 2007]. Here we summarize only some of the related research problems.

One of the well-studied related problems is that of Top-K retrieval in probabilistic databases. There, one of the goals is to *efficiently* compute the top-K answers for a given K , without computing the exact probability for each object to be in the answer set [Soliman et al. 2007; Re et al. 2007; Zhang and Chomicki 2009; Cormode et al. 2009], since these computations can be expensive. However, the primary focus of our work is the *quality*, and not the efficiency, angle of the problem. There is also a large number of efforts on top-K retrieval on deterministic databases [Theobald et al. 2004; Chang and Hwang 2002; Chaudhuri et al. 2004]. However, the scope of our work is quite different from that.

Another related problem is that of automatic keyword selection for indexing. There, the goal is to select the best keywords that define the document, because the adequacy of the indexing determines the quality of the system in responding the user queries. All the keywords in the document can be used for indexing, but in that case although the indexing is exhaustive, it is not specific. Thus it leads to

high recall, but low precision [Harter 1975; Bookstein and R.Swanson 1975].

Kraft [Kraft 1973], formulated the relevance between documents and queries using decision theory. For every possible keyword a decision is made whether that keyword is relevant to the given document. Hence, the relevance has mutually exclusive alternatives and decision theory can be used for that. Using decision theory and 2-Poisson processes Harter [Harter 1975] proposed an algorithm to define a measure for indexability by minimizing the cost, where cost is defined as the user satisfaction. The algorithm is built upon the study in [Bookstein and R.Swanson 1975], and aims to optimize the expected precision for each expected recall level. It uses the Probability Ranking Principle explained in Robertson [Robertson 1977], which states that for the optimum performance on a given query a document retrieval system should rank the documents in order of their probability of relevance to the query. The main difference between this algorithm and ours is that we optimize the query results using the expected F_α .

In Nottelmann and Fuhr [Nottelmann and Fuhr], similar expected precision and recall measures are used to decide which resource to select for further querying. Three different methods are proposed for estimating retrieval quality in resource selection. In Takenobu et al. [Takenobu et al. 2002], similar effectiveness measures are used for index term selection, when the ground truth for the queries are known. The algorithm first ranks the terms using scores and uses certain term effectiveness measures to decide when to stop adding index terms. Two different term effectiveness measures are defined and used: term precision and term F-measure. A cut-off value for term selection can be determined when the index term added to the query maximizes the quality measure: either Precision or F-measure. The keywords are selected for document indexing via a supervised learning method.

Work by Li and Deshpande has been developed concurrently with ours, and studies the problem of finding answers that are optimal, in the expected sense under Jaccard and other quality metrics, on top of the probabilistic And/Xor tree representation, e.g. [Li and Deshpande 2009]. They propose a theoretically elegant algorithm that could be applied to the problem of object selection after the probabilities $\mathbb{P}(O_i \in G_Q)$ are computed. That algorithm can be adapted to handle F_α measure, but has $O(n^3 \log n)$ complexity, and thus infeasible in our problem settings since n can be large. In this paper we propose an algorithm that instead of spending computational resources on computing the expected values exactly, estimates these values quickly, resulting in a linear time complexity algorithm that reaches virtually the same quality as the non-estimate based solution.

7. CONCLUSIONS AND FUTURE WORK

In this paper we studied the problem of maximizing the quality of selection queries on probabilistically annotated content. Two types of queries have been studied: attribute value- and object- selection queries. Several algorithms for answering such queries have been proposed. We have demonstrated that the proposed solutions significantly outperform known techniques across variety of datasets and scenarios. As future work, we plan to develop similar techniques for different types of queries in the context of generic probabilistic databases.

REFERENCES

- ANTOVA, L., JANSEN, T., KOCH, C., AND OLTEANU, D. 2008. Fast and simple relational processing of uncertain data. In *ICDE*.
- ASHISH, N., MEHROTA, S., AND PIRZADEH, P. 2009. XAR: An integrated framework for free text information extraction. In *IEEE CSIE Conference*.
- ASUNCION, A., SMYTH, P., AND WELLING, M. 2008. Asynchronous distributed learning of topic models. In *NIPS*.
- BAEZA-YATES, R. AND RIBERTO-NETO, B. 1999. *Modern Information Retrieval*. Addison-Wesley.
- BOOKSTEIN, A. AND R. SWANSON, D. 1975. A decision theoretic foundation for indexing. *JASIS*.
- CARROLL, J. AND BRISCOE, T. 2002. High precision extraction of grammatical relations. In *COLING*.
- CHANG, K. AND HWANG, S. 2002. Minimal probing: supporting expensive predicates for top-k queries. In *SIGMOD*.
- CHAUDHURI, S., GANJAM, K., GANTI, V., AND MOTWANI, R. 2003. Robust and efficient fuzzy match for online data cleaning. In *ACM SIGMOD Conference*.
- CHAUDHURI, S., GRAVANO, L., AND MARIAN, A. 2004. Optimizing top-k selection queries over multimedia repositories. *TKDE*.
- CHEN, J., TAN, T., AND MULHEM, P. 2001. A method for photograph indexing using speech annotation. In *IEEE Pacific Rim Conference on Multimedia*.
- CHEN, S., KALASHNIKOV, D. V., AND MEHROTRA, S. 2007. Adaptive graphical approach to entity resolution. In *Proc. of ACM IEEE Joint Conference on Digital Libraries (JCDL 2007)*. Vancouver, British Columbia, Canada.
- CHEN, Z. S., KALASHNIKOV, D. V., AND MEHROTRA, S. 2009. Exploiting context analysis for combining multiple entity resolution systems. In *Proc. of ACM SIGMOD International Conference on Management of Data (ACM SIGMOD 2009)*. Providence, RI, USA.
- CHENG, R., KALASHNIKOV, D. V., AND PRABHAKAR, S. 2003. Evaluating probabilistic queries over imprecise data. In *Proc. of ACM SIGMOD International Conference on Management of Data (ACM SIGMOD 2003)*. San Diego, CA, USA.
- CHENG, R., KALASHNIKOV, D. V., AND PRABHAKAR, S. 2007. Evaluation of probabilistic queries over imprecise data in constantly-evolving environments. *Information Systems Journal* 32, 1 (Mar.), 104–130.
- CORMODE, G., LI, F., AND YI, K. 2009. Semantics of ranking queries for probabilistic data and expected ranks. In *ICDE*.
- DALVI, N. AND SUCIU, D. 2004. Efficient query evaluation on probabilistic databases. In *VLDB*.
- DESAI, C., KALASHNIKOV, D. V., MEHROTRA, S., AND VENKATASUBRAMANIAN, N. 2009. Using semantics for speech annotation of images. In *Proc. of the 25th IEEE Int'l Conference on Data Engineering (IEEE ICDE 2009)*. Shanghai, China. short publication.
- HARTER, S. 1975. A probabilistic approach to automatic keyword indexing: Part II, An algorithm for probabilistic indexing. *JASIS*.
- HERNANDEZ, M. AND STOLFO, S. 1995. The merge/purge problem for large databases. In *ACM SIGMOD Conference*.
- KALASHNIKOV, D. V., MA, Y., MEHROTRA, S., AND HARIHARAN, R. 2006. Index for fast retrieval of uncertain spatial point data. In *Proc. of Int'l Symposium on Advances in Geographic Information Systems (ACM GIS 2006)*. Arlington, VA, USA.
- KALASHNIKOV, D. V. AND MEHROTRA, S. 2006. Domain-independent data cleaning via analysis of entity-relationship graph. *ACM Transactions on Database Systems (ACM TODS)* 31, 2 (June), 716–767.
- KALASHNIKOV, D. V., MEHROTRA, S., AND CHEN, Z. 2005. Exploiting relationships for domain-independent data cleaning. In *SIAM International Conference on Data Mining (SIAM Data Mining 2005)*. Newport Beach, CA, USA.
- KALASHNIKOV, D. V., MEHROTRA, S., XU, J., AND VENKATASUBRAMANIAN, N. 2011. A semantics-based approach for speech annotation of images. *IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE)* 23, 9 (Sept.), 1373–1387.
- ACM Transactions on Database Systems, Vol. V, No. N, August 2011.

- KRAFT, D. 1973. A decision theory view of the information retrieval situation: An operations research approach. *JASIS*.
- LI, J. AND DESHPANDE, A. 2009. Consensus answers for queries over probabilistic databases. In *PODS*.
- MA, Y., KALASHNIKOV, D. V., AND MEHROTRA, S. 2008. Towards managing uncertain spatial information for situational awareness applications. *IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE)* 20, 10 (Oct.).
- MARTÍN-BAUTISTA, M. J., SÁNCHEZ, D., MIRANDA, M. A. V., AND LARSEN, H. L. 2000. Measuring effectiveness in fuzzy information retrieval. In *FQAS*.
- MENESTRINA, D., BENJELLOUN, O., AND GARCIA-MOLINA, H. 2006. Generic entity resolution with data confidences. In *CleanDB*.
- MOENCK, R. T. 1976. Practical fast polynomial multiplication. In *ACM ISSAC*.
- NICULESCU-MIZIL, A. AND CARUANA, R. 2005. Predicting good probabilities with supervised learning. In *ICML*.
- NOTTELMANN, H. AND FUHR. Evaluating different methods of estimating retrieval quality for resource selection. In *SIGIR'03*.
- NURAY-TURAN, R., KALASHNIKOV, D. V., AND MEHROTRA, S. 2007. Self-tuning in graph-based reference disambiguation. In *Proc. of the 12th International Conference on Database Systems for Advanced Applications (DASFAA 2007)*, Springer LNCS. Bangkok, Thailand.
- RAVINDRA, G., BALAKRISHNAN, N., AND RAMAKRISHNAN, K. R. 2004. Automatic evaluation of extract summaries using fuzzy f-score measure. In *Fifth International Conference on Knowledge Based Computer Systems (KBCS 2004)*.
- RE, C., DALVI, N. N., AND SUCIU, D. 2007. Efficient top-k query evaluation on probabilistic data. In *ICDE*.
- ROBERTSON, S. E. 1977. The probability ranking principle in IR. *Journal of Documentation*.
- SARMA, A. D., THEOBALD, M., AND WIDOM, J. 2008. Exploiting lineage for confidence computation in uncertain and probabilistic databases. In *ICDE*.
- SATPAL, S. AND SARAWAGI, S. 2007. Domain adaptation of conditional probability models via feature subsetting. In *PKDD*.
- SINGH, S., MAYFIELD, C., MITTAL, S., PRABHAKAR, S., HAMBRUSCH, S. E., AND SHAH, R. 2008. The orion uncertain data management system. In *COMAD*. 273–276.
- SOLIMAN, M. A., ILYAS, I. F., AND CHENG, K. C.-C. 2007. Top-k query processing in uncertain databases. In *ICDE*.
- STEYVERS, M., SMYTH, P., ROSEN-ZVI, M., AND GRIFFITHS, T. L. 2004. Probabilistic author-topic models for information discovery. In *KDD*. 306–315.
- TAKENOBU, T., KENJI, K., HIRONORI, O., AND HOZUMI, T. 2002. Selecting effective index terms using a decision tree. *NLE*.
- THEOBALD, M., WEIKUM, G., AND SCHENKEL, R. 2004. Top-k query evaluation with probabilistic guarantees. In *VLDB*.
- WICK, M. L., ROHANIMANESH, K., SCHULTZ, K., AND MCCALLUM, A. 2008. A unified approach for schema matching, coreference and canonicalization. In *KDD*.
- WIDOM, J. 2005. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR*. 262–276.
- ZADROZNY, B. AND ELKAN, C. 2001. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *ICML*. 609–616.
- ZADROZNY, B. AND ELKAN, C. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *SIGKDD*.
- ZHANG, J. AND YANG, Y. 2004. Probabilistic score estimation with piecewise logistic regression. In *ICML*.
- ZHANG, X. AND CHOMICKI, J. 2009. Semantics and evaluation of top-k queries in probabilistic databases. *DPD*.
- ZIOLKO, B., MANANDHAR, S., AND WILSON, R. 2007. Fuzzy recall and precision for speech segmentation evaluation. In *Proceedings of 3rd Language and Technology Conference*.