Inferencing Underspecified Natural Language Utterances in Visual Analysis

Vidya Setlur[*]
Melanie Tory[†]
Alex Djalali[‡]
Tabeau Software
Palo Alto, California 94306

ABSTRACT

Handling ambiguity and underspecification of users' utterances is challenging, particularly for natural language interfaces that help with visual analytical tasks. Constraints in the underlying analytical platform and the users' expectations of high precision and recall require thoughtful inferencing to help generate useful responses. In this paper, we introduce a system to resolve partial utterances based on syntactic and semantic constraints of the underlying analytical expressions. We extend inferencing based on best practices in information visualization to generate useful visualization responses. We employ heuristics to help constrain the solution space of possible inferences, and apply ranking logic to the interpretations based on relevancy. We evaluate the quality of inferred interpretations based on relevancy and analytical usefulness.

CCS CONCEPTS

• **Human-centered computing** Interaction techniques; Text input.

KEYWORDS

natural language interface, visual analysis, inferencing

ACM Reference Format:
Vidya Setlur, Melanie Tory, and Alex Djalali. 2019. Inferencing Underspecified Natural Language Utterances in Visual Analysis. In *24th International Conference on Intelligent User Interfaces (IUI '19), March 17–20, 2019, Marina del Ray, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3301275.3302270

1 INTRODUCTION

Ambiguity and underspecification in language are prevalent in any form of communication, but is not necessarily an issue with human-to-human interaction [5]. Humans are adept at disambiguation by clarifying and repairing utterances. The issue of underspecificity in language transcends to computer systems, particularly web search systems. Research shows that queries often come out ambiguous or underspecified, especially with the increase of mobile device usage in recent years [28]. Similarly, supporting natural language interaction with visual analytical systems is often challenging. First, users tend to type utterances that are linguistically simple and underspecified, while the visual analytics system has more complicated nuances of realizing these utterances against the underlying data and analytical functions. Second, users expect high precision and recall from such natural language interfaces. Unlike web search systems relying on document indexing, visual analytical systems are constrained by the underlying analytical engine and data characteristics.

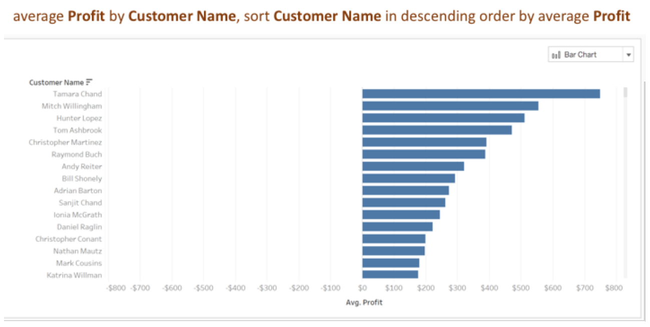
A 'smart' system can attempt to effect a match between the concepts in the utterances and the concepts known by the system. While follow-up repair utterances can help resolve ambiguities that a natural language interface may encounter, such systems are often constrained by the domain of the knowledge base or context in which the interaction occurs. In addition, analytical concepts may not map directly from utterances to the underlying information. For example, a user may be looking at housing data in a particular neighborhood, and type in an utterance, "show me homes with good schools and at least 2 bedrooms." If the concept of *good* is not present in the underlying data, then the user's intent is rather ambiguous. The system would need to infer *good* to be perhaps a low crime rate, good school ratings, or a great walking score.

A successful user interaction may involve refining the utterance based on results and reformulating the query strategy if the utterance is too broad, narrow, or misformulated. While repair through follow-up utterances or employing a mixed initiative approach [12, 17, 34] is prevalent, inferencing can help determine the mapping of the intent posed by terms used in these natural language utterances, and provide sensible defaults to the user [23]. Most intelligent interface designs assume to one degree or another that the user will be "kept in the loop" to negotiate with the system, resolving ambiguities, making relevancy judgements, and revising searches based on feedback provided by the system [36].

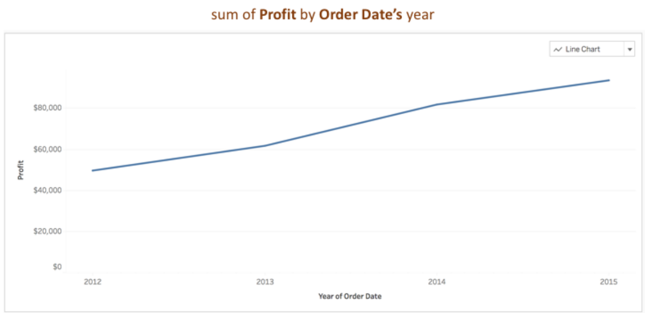
The tendency for users to employ underspecified utterances in data analytics was recognized as early as 2010 by Grammel *et al.* [14] and has been similarly observed in other domains (*e.g.* [25]). A recent Wizard Of Oz study that we conducted (unpublished) reinforced the need for inferencing to support underspecification. We found that nearly all input utterances were underspecified in some way. For instance, visual encodings were nearly always incomplete or left out, and were often specified indirectly as an analytical goal

[*]vsetlur@tableau.com
[†]mtory@tableau.com
[‡]adjalali@tableau.com

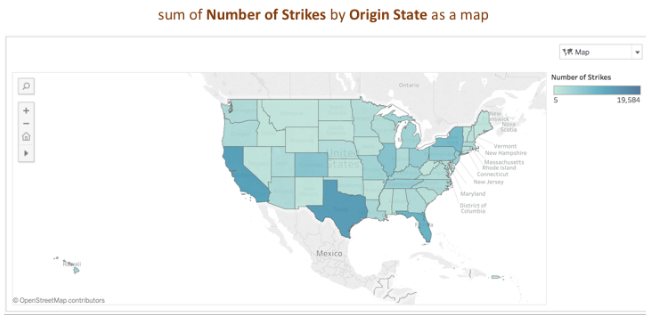
IUI '19, March 17–20, 2019, Marina del Ray, CA, USA
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6272-6/19/03...$15.00
https://doi.org/10.1145/3301275.3302270


average **Profit** by **Customer Name**, sort **Customer Name** in descending order by average **Profit**

(a) "sort customers by their profit"


sum of **Profit** by **Order Date's** year

(b) "what's my profit over time?"


sum of **Number of Strikes** by **Origin State** as a map

(c) "show me all the strikes on a map"

Figure 1: System inferences made for underspecified utterances during visual analysis. (a): Infers a descending sort order to sort *Customer Names* by *Profit*. (b): A user expresses an intent of 'time' that is resolved to a time attribute *Order Date* aggregated at its year level. (c): Infers a geographical attribute, *State*, in order to display the number of bird strikes in a map employing user popularity data and heuristics for analytical usefulness. Note: The canonical form is displayed on top of each figure for legibility while the input utterance is shown in the caption.

(*e.g.* 'correlation'). Here we introduce an inferencing system for handling underspecification, grounded in our empirical data of language cues that indicate intent as well as expected behavior.

2 RELATED WORK

Inferencing underspecified queries in web search
Underspecified utterances and ambiguity are prevalent in web search systems - whether the large inventory of documents indexed by search, the esoteric collections in niche engines, or simply because users make assumptions of what the search system interprets from their queries. One challenge is the problem of irrelevant search results. Such search results usually arise due to ambiguous queries or semantic mismatches.

Search engines have utilized the notion of context to help restrict the ambiguity space. By processing searches in the context of the information surrounding the queries, search results can better reflect the user's actual intentions [6]. IntelliZap selects important context words and performs word sense disambiguation to prepare a set of augmented queries for subsequent search [1]. Sajjad *et al.* proposed a question generation system for handling underspecified queries from domain-specific search engines [30].

Search diversification is another technique employed by web search to satisfy as many intents as possible behind an underspecified query [7]. When a search engine receives an ambiguous query and has limited knowledge about the user's intent, the system can present a diversified result list that covers several interpretations of the query. Diversification algorithms explore modeling the probabilistic knowledge of user intent, document classification, and how many relevant documents a user will require to maximize the probability of satisfaction when posing ambiguous queries [32, 44]. Other research focuses on optimizing search result presentation for queries with diverse user intent by selectively presenting query suggestions for leading users to more relevant search results [2, 3].

Another cause for irrelevant results is the 'one-size-fits-all' approach where an identical query from different users in different contexts generates the same set of results. To improve search relevancy, systems have focused on generating a personalized search result list based on based on the user's profile [27, 39]. Search personalization can also employ machine learning from human relevance judgments such as click-through interactions [37]. Other approaches perform web query disambiguation based on predictions from short glimpses of user search activity using statistical learning models [24].

Inferencing underspecified queries in visual analytics
Similar to web search systems, natural language interfaces for visual analysis need to be able to interpret a broad range of utterances to keep users in their analytical workflows. DataTone inferred a user's intent, producing a chart according to that inference, and then providing ambiguity widgets through which the user could adjust the system's default choice [12]. Eviza and Analyza supported simple pragmatics in analytical interaction through contextual inferencing, wherein context established by the preceding dialog was used to create a complete utterance [11, 34]. Evizeon [17] and Orko [38] extended the notion of pragmatics in analytical conversation by using the knowledge of data attributes, values, and data related expressions. Ambiguity was handled with targeted textual feedback and ambiguity widgets. Inferencing however, was limited to only *filtering* in the analytical workflow.

Our work focuses specifically on enhancing the notion of inferencing for underspecified utterances during an analytical workflow. Leveraging semantic and syntactic constraints posed by VizQL [40] as well as applying heuristics from best practices in the field of information visualization, we can support a richer repertoire of analytical expressions.

3 CONTRIBUTIONS

In this paper we introduce a natural language system for resolving ambiguity and underspecification in natural language utterances. Our system infers data attributes and appropriate visualizations that help satisfy the intent in the utterances. In Figure 1a, our system infers a descending sort order, matching typical user expectations that the highest value will be at the top. Our algorithm can also make sensible inferencing to map an abstract concept such as 'time' to an appropriate time attribute (Figure 1b). Further, with data involving hierarchies such as time and geography, the system can infer data attributes at an appropriate level of detail, ranking more salient geographical attributes such as 'State' over inferring a 'Postal Code' (Figure 1c).

The contributions of this paper can be summarized as follows:

- We introduce a lightweight, intermediate language, *Arklang* to resolve natural language utterances to formal queries that can be executed against a visual analytics system.
- We describe a set of inferencing techniques based on syntactic and semantic constraints of partial analytical expressions and to handle vague concepts. We also support visualization responses explicitly requested by users through inferring meaningful data attributes.
- A relevancy metric is employed to identify a set of valid inferences based on data popularity, the constraints of the analytical system as well as drawing from principles of effective graphical data presentation.
- The system provides a set of interaction widgets to provide other relevant options to the users for overriding and clarifying inferencing defaults.
- We validated the appropriateness of our inferencing approach through a survey; respondants assessed the quality of visualization responses for a diverse set of underspecified input queries obtained from log data collected in a two month deployment of our system.

4 SYSTEM OVERVIEW

Conventional query languages such as SQL are powerful from a mathematical perspective, but are not conducive for natural language parsing. This is because users' input utterances tend to be more colloquial with less semantic and syntactic rigor when compared to a database query language. Implementing natural language interfaces for databases that can map user utterances to a query language can thus be challenging. The grammar rules underlying their syntax must be coerced into Chomsky Normal Form [9] to be available for standard natural language parsing techniques. While statistical and machine learning techniques can be employed, manually authoring and tuning a grammar for each new database is brittle and prohibitively expensive. In addition, to parse questions posed to a particular database, the statistical parser would need to be trained on a corpus of questions specific to that database. Several systems have developed natural language interfaces to databases [4, 15, 16, 20, 26]. Such systems introduce semantically tractable sentences that can be translated to a unique semantic interpretation by analyzing lexicons and semantic constraints of the underlying databases. Inspired by this previous work, we implement a lightweight, intermediate language, *ArkLang*, to represent an intermediate logical query generated from an utterance, focusing specifically on visual analysis tasks based on the semantic constraints of VizQL [40].

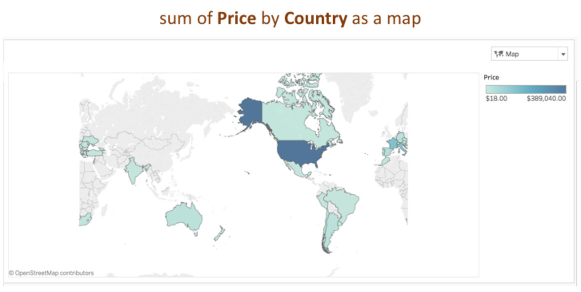
ArkLang is designed to resolve natural language utterances to formal queries that can be executed against a visual analytics system. We lexically translate the natural language utterance into ArkLang, and then compile ArkLang into a series of instructions employing a visualization query language, VizQL [40] to issue a query against a database. VizQL is a formal language for describing tables, charts, graphs, maps, time series and tables of visualizations. These different types of visual representations are unified into one framework, coupling query, analysis and visualization. This declarative language facilitates transformation from one visual representation to another (*e.g.* from a list view to a cross-tab to a chart).

In order to perform a lexical translation from natural language into ArkLang, we require a *lexicon* mapping natural language words to their ArkLang concepts, which in turn are mapped to their corresponding VizQL counterparts. To help with inferencing and choosing salient attributes and values, we leverage a *Semantic Model* (SM) derived directly from the underlying database. The SM represents the database schema and contains metadata about attributes, such as alternative labels, or *synonyms*. Various data types (*i.e.*, 'text,' 'date,' 'geospatial,' 'boolean,' and 'numeric') and attribute semantics are stored, such as currency type (*e.g.* United States Dollar) and semantic role (*e.g.* 'City' role for a geospatial attribute). Statistical values such as the data distribution, range limits, average, cardinality are also captured for each attribute. The Semantic Model is augmented with a set of analytical concepts found in many query languages (*e.g.,* average, filter, sort). It also distinguishes between attributes that are *measures* (*i.e.* attributes that can be measured, aggregated, or used for mathematical operations) and *dimensions* (*i.e.* fields that cannot be aggregated except as count). Popularity scores are also computed and associated with each attribute, indicating how often the attribute is used in visualizations generated from the datasource.

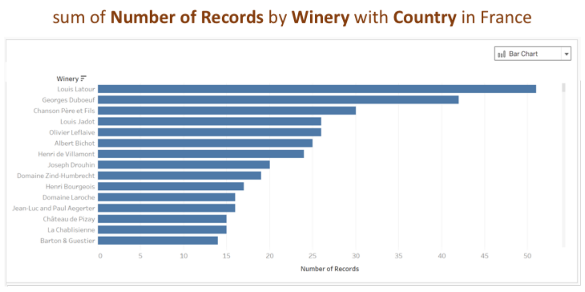
4.1 Formal Representation

ArkLang describes a formal query language that can be generated from a set of Semantic Models representing their corresponding databases, a context free grammar (CFG) [9], and a set of semantic constraints. By a *dialect* of ArkLang, we mean the set of all syntactically valid and semantically meaningful *analytical expressions* that can be obtained by fixing a particular SM and leveraging our fixed CFG and fixed set of semantic constraints. We use a Cocke-Kasami-Younger (CKY) parsing algorithm that employs bottom-up parsing and dynamic programming on CFGs [19]. The input to the underlying CKY parser is this context-free grammar with production rules augmented with both syntactic and semantic predicates based on analytical expressions that correspond to five basic database operations found in VizQL [40] and are shown in Figure 2. The analytical concepts in ArkLang are defined below:

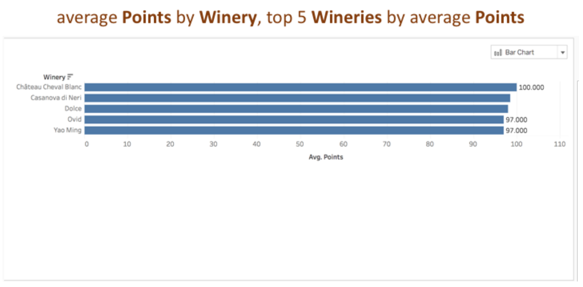
- *Fields* are a finite set of database attributes. *E.g.*, 'sales', 'product category.'
- *Values* are a finite set of database values. *E.g.*, '$100', 'Avery Leckman.'
- *Aggregations* are a finite set of operators where the values of multiple rows are grouped together to form a single value


sum of Price by Country as a map
Map
Price
$18.00
$389,040.00
© OpenStreetMap contributors

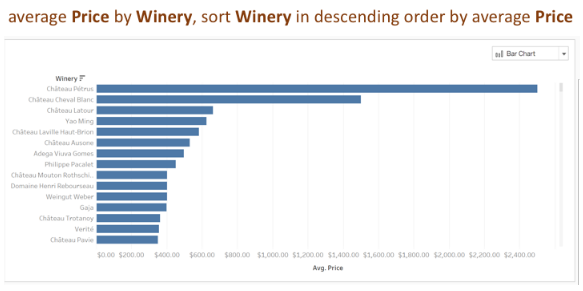
(a) "what's the sum of price for each country?"



(b) "wineries in france"



(c) "top 5 wineries by average points"



(d) "sort wineries by average price"

Figure 2: Five basic analytical expressions supported in ArkLang. (a) An aggregation expression "sum of Price" with a group expression "by Country", (b) A filter expression "in France", (c) A limit expression "top 5 Winery", (d) A sort expression "sort Winery".

based on a mathematical operation. *E.g.*, 'average', 'median', 'count', 'distinct count.'

- *Groups* are a finite set of operators that partition the data into categories shown in a data visualization. *E.g.*, 'by' a field.
- *Filters* are a finite set of operators that return a subset of the field's domain. *E.g.*, 'filter to', 'at least', 'between', 'at most.'
- *Limits* are a finite set of operators akin to *Filters* that return a subset of the field's domain, restricting up to n rows, where $1 \leq n \leq N$. N is the total number of rows in the domain. *E.g.*, 'top', 'bottom.'
- *Sorts* are a finite set of operators that arrange data rows in an order, *i.e.*, 'ascending', 'descending', 'alphabetical.'

With these analytical concepts defined in ArkLang, we can support various analytical *expressions*. To address the problem of proliferation of ambiguous syntactic parses inherent to natural language querying, our system assigns canonical representations to these analytical expressions [10]. These canonical structures are unambiguous from the point of view of the parser and our system is able to choose quickly between multiple syntactic parses. For example, instead of enumerating all possible phrases, the grammar returns a deterministic output for a given input utterance. Here is the set of basic analytical expressions, along with their canonical forms in the ArkLang dialect with examples shown in Figure 2. Note that the canonical forms are shown at the top of each figure for legibility reasons with the input utterances in their corresponding captions.

- Aggregation expression: If $agg \in Aggregations$ and att is an *Attribute* with the canonical form `[agg att]`; (*e.g.*, "average Sales" where 'average' is agg, 'Sales' is att).
- Group expression: If $grp \in Groups$ and att is an attribute with the canonical form `[grp att]`; (*e.g.* "by Region" where 'by' is grp, 'Region' is att).
- Filter expression: If att is an attribute, $filter \in Filters$, and $val \in Values$ with the canonical form `[att filter val]`; (*e.g.*,"Customer Name starts with John" where 'Customer' is att, 'starts with' is $filter$, 'John' is val).
- Limit expression: If $limit \in Limits$, $val \in Values$, $ge \in$ *group expressions*, and $ae \in$ aggregation expressions with the canonical form `[limit val ge ae]`; (*e.g.*, "top 5 Wineries by sum of Sales" where 'top' is $limit$, '5' is val, 'Wineries' is the attribute to group by, 'sum of Sales' is the *aggregation expression*).
- Sort expression: If $sort \in Sorts$, $ge \in$ group expressions, and $ae \in$ *aggregation expressions* with the canonical form `[sort ge ae]`; (*e.g.*, "sort Products in ascending order by sum of Profit" where 'ascending order' is the *sort*, 'Products' is the attribute to group by, 'sum of Profit' is the *aggregation expression*).

In VizQL, expressions referencing unaggregated datasource columns are computed for each row in the underlying table [40]. In this case, the dimensionality of the expression is *row-level*. Expressions referencing aggregated data source columns are computed at the dimensionality defined by the dimensions in the view. In this case, the dimensionality of the expression is *view-level*. ArkLang supports these levels of detail with the concept of an attribute and a corresponding aggregation expression. Such embeddings allow us to ask higher-order analytical questions involving level of detail. ArkLang's recursive structure allows us to formulate filtration conditions at multiple levels of detail, supporting filtration at both row- and view-levels. Let us assume that our data contains a *Sales* attribute with underlying numeric values. ArkLang is expressive enough to formulate row-level filter expressions like "sales at least $100." Similarly, "by country, average sales at least $100" is a view-level filtration condition that removes any sets of rows grouped by the *Country* attribute whose average per the *Sales* attribute is not greater than $100.

The translation from natural language input to VizQL commands for generating a visualization response, is implemented in Algorithm 1.

Algorithm 1: Natural language translation to VizQL

```
Input: natural language token
Output: VizQL
Let $f$ be a translation function mapping a natural language
  word, e.g. into an ArkLang concept 'average'.
Let $g$ be (a top-down recursive) translation function mapping
  analytical expressions of ArkLang to VizQL.
Then $h$ is defined as the composition of $f$ and $g$ mapping a
  natural language into VizQL.
1 Perform a lexical translation from natural language into
    ArkLang, e.g., f(mean) = f(avg) = average and f(wine
    prices) = Price.
2 Leverage the CFG and a set of grammar rules to parse the
    resultant translated terms into ArkLang dialect,
    $average \in Aggregations$ and $wine\ prices \in Fields$, with
    $average, Price \in aggregation\ expressions$.
3 Compile the ArkLang sentences into VizQL commands and
    issue those commands against a database, e.g. g([average,
    Price]).
```

5 INFERENCING LOGIC

Handling underspecification in natural language utterances involves addressing their ambiguities and making thoughtful choices as to what the user's intent may be. By leveraging the *known* syntactic and semantic structures in the analytical expressions defined in ArkLang, we apply a set of inferencing rules to address missing information. Our algorithm handles four types of inferencing: (a) handling underspecification *within* each of the 5 analytical expressions, *i.e.* intra-phrasal inferencing, (b) handling underspecification *between* the analytical expressions, *i.e.* inter-phrasal inferencing, (c) handling underspecification in the analytical expressions when a user explicitly specifies a visualization type that she would like to see the analytical response represented in, and (d) inferring reasonable defaults for vague underspecified concepts such as 'expensive' and 'popular.' We explore each of these inferencing types in the subsequent sections.

A common issue with inferring missing detail is that the algorithm could result in multiple valid matches creating a confusion set. When this set is large, it is difficult for a user to navigate through the matches to find her target interpretation. We apply a notion of *relevancy* to narrow down the possible semantic representations for each syntactic analysis, and derive a subset of plausible interpretations from that confusion set. Relevancy is based on string similarity scores between the input natural language utterance and the interpretations returned by our system. We leverage an off-the-shelf term frequency-inverse document frequency (TFIDF) [31] similarity score in Elasticsearch[1] as the scoring is easy to compute and tends to work well for our purpose. For utterances where the fields are missing (*e.g.* an underspecified filter expression "over $200"), our system will infer the missing field based on a popularity score from usage data and how often the field was used in visualizations on the datasource. Finally, we identify a set of heuristics based on principles of information visualization to further prune the visualization responses for their analytical usefulness [8, 29]. For example, when inferring a time concept in an utterance such as "show me orders 2015", relative time concepts (*e.g.* "last 2015 years") tend to be less salient than absolute time concepts (*e.g.* "in the year 2015").

5.1 Repair and refinement

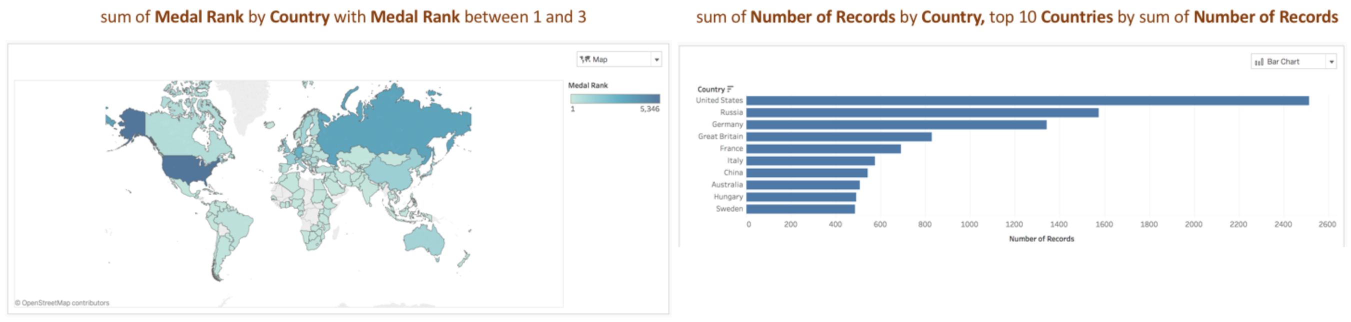
With natural language interfaces, understanding and interpreting user intent is always a challenging problem. It is pertinent for systems making smart defaults and inferences about user intent to have provisions for repair and refinement during the interaction experience [33]. We draw inspiration from other natural language interfaces that support handling repair [11, 12, 17, 34, 38] through similar refinement widgets. Such systems tend to be more useful when they not only parse the linguistic structure of the utterances, but also effectively address inevitable ambiguity in inferencing through repair utterances, feedback and ambiguity widgets.

The intent for refinement interaction is to provide a complementary model for users using our natural language system to observe the analytical operations the system performs after interpreting an analytical utterance (*e.g.* "low sales" will be interpreted as "sales between X and Y," where X and Y are numerical limits). A user can then repair or correct an interpretation if the inferencing is not optimal. The refinement widgets also provide a means for data and feature discoverability. *E.g.*, after typing "maximum sales" a user could see other aggregation functions in the set like *sum*, *average* as well as other numerical attributes with similar data properties such as *Profit* and *Discount*. We discuss in more detail how refinement is supported in our system in Section *Resolving vague predicates* and in Figure 6.

5.2 Intra-phrasal inferencing

Our inferencing logic relies on constraints imposed by the syntactic and semantic structure of the various expressions. Each *fully specified expression* has a known set of parameters, with a known set of constraints determining the natural language tokens, fields and values. For example, in Figure 3a, the user specifies a range of numbers without mentioning the type of filter. Based on the grammar rules for filter expressions, our system infers a filter 'between' to generate a fully specified '*filter expression*. Given that these natural language utterances need to resolve into VizQL queries [40], we apply additional constraints to generate viable visualizations in Tableau. Specifically, filter and limit expressions require a group or an aggregation expression to be present. A limit expression also requires an aggregated

[1]`https://www.elastic.co/`



(a) "countries with rank 1 3"

(b) "top 10 countries"

Figure 3: Intra-phrasal inferencing. (a) An underspecified filter expression where the filter type is an open variable. The system infers the filter *between* and generates an appropriate response with a canonical output "by Country with Medal Rank between 1 and 3." (b) An underspecified limit expression where the aggregation expression is an open variable. The system infers 'sum of Number of Records' showing the number of records in the view. A fully specified limit expression "top 10 Country by sum of Number of Records." is generated to limit the dimension 'Country'.

expression to limit the attribute by. A sort expression requires the dimension that is sorted upon, to be in a group expression. We established a rule that infers `SUM(NumberOfRecords)` when a user does not specify an aggregation expression. 'Number of Records' is an automatically generated calculated field in Tableau that contains value 1, associated with each record in the database. In Figure 3b, a fully specified *limit expression* requires a numerical attribute with a suitable aggregation function applied, which is "sum of Number of Records" in this case.

For each of the analytical expressions, let us assume a finite set of variables of that type. For example, for the group expression, the variables would be $g_1, ..., g_n$ for $n \leq \omega$. We identify an expression to be underspecified if the expression contains at least one *free* variable. For example, an underspecified aggregation expression would be of the form **average**,x, where x is a *Field* variable. While the *Aggregation*, **average**, in this expression is defined, its *Field* is not—it is the free variable x. Similarly, **sales**, **at least**,y, is an underspecified filter expression where y is a *Value* variable.

We can now reify the notion of what we call *intra-phrasal inferencing* as it relates to ArkLang: It is the process of instantiating an open variable in an ArkLang expression with an actual instance of that data or concept type. Let us refer to such a process as the function *Intra* and defined, in part, ostensively as follows.

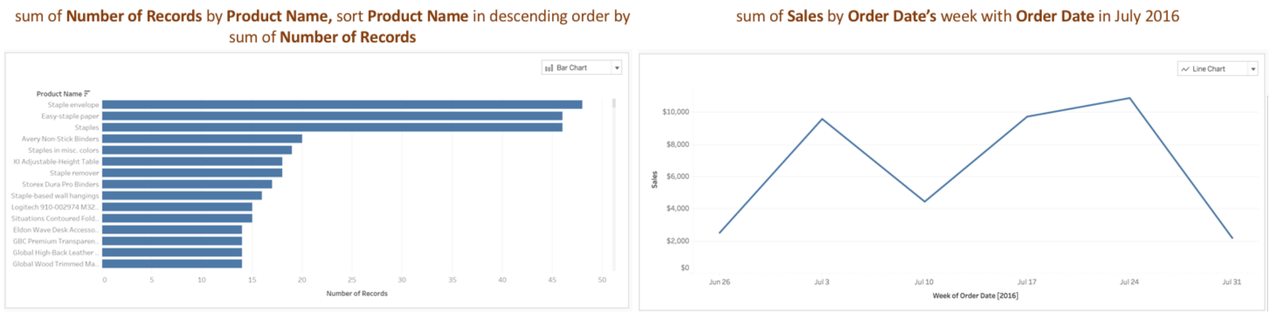
- If [**average**, x] is an underspecified *aggregation expression* and x is a free variable of type *Field*, then *Intra*([**average**, x]) = [**average**,**sales**] is a fully specified *aggregation expression*.
- If [x, **country**] is an underspecified *group expression* and x is a free variable of type *Group*, then *Intra*([x, **country**]) = [**by**, **country**] is a fully specified *group expression*.
- If [**sales**, **at least**, x] is an underspecified *filter expression* and x is a free variable of type *Value*, then *Intra*([**sales**, **at least**, x]) = [**average**, **sales**, **$100**] is a fully specified *filter expression*.
- If [x, **10**, **by**, **country**, **average**, **sales**] is an underspecified *limit expression* and x is a free variable of type *Limit*, then *Intra*([x, **10**, **by**, **country**, **average**, **sales**]) = [**top**, **10**, **by**, **country**, **average**, **sales**] is a full specified *limit expression*.
- If [**by**, **country**, x, **average**, **sales**] is an underspecified *sort expression* and x is a free variable of type *Sort*, then *Intra*([**by**, **country**, x, **average**, **sales**]) = [**by**, **country**, **descending**, **average**, **sales**] is a full specified *sort expression*.

The function *Intra* selects the non-logical constant of the appropriate type to instantiate for x based on relevancy. We then compute the top n most relevant non-logical constants and instantiate them for x, resulting in a set of n fully specified analytical expressions.

5.3 Inter-phrasal inferencing

Given a fully specified analytical expression of ArkLang, we infer additional fully specified analytical expression either because (*i*) the underlying query language we compile ArkLang into, *i.e.*, VizQL, requires such additional expressions to be co-present for purposes of query specification or (*ii*) such additional expressions improve the analytical usefulness of the resultant visualization. Inter-phrasal inferencing infers these various constraints between these analytical expressions.

In regard to (*i*), the visual specification for VizQL requires either measure fields to be aggregated or dimension fields grouping the data into panes to generate a visualization. Therefore filter and limit expressions require aggregated measures and/or grouped dimensions in play to select subsets of the data for analysis. A sort expression has a stricter constraint that requires the dimension that is being sorted to also be used to group the data in the visualization. Consider the natural language representation of a sort expression "sort Countries in descending order by average Sales." In order to compile this expression into VizQL, the underlying attribute of this expression 'Country' requires itself understood as being the group expression 'by Country,' to be compiled in conjunction with the sort expression. So, when our system encounters any sort expression, if its underlying group expression does not appear with it conjunctively, our system must introspect the sort expression, retrieve that group expression,



(a) "sort product from biggest"

(b) "sales throughout july 2016"

Figure 4: Inter-phrasal inferencing. (a) A sort expression requires the dimension that is being sorted to also be used to group the data in the visualization. Here 'by Product' is inferred as a group expression and we also infer 'sum of Number of Records' to generate an analytically useful bar chart. (b) We infer a group expression 'by Order Date's week', which is 1 level lower than the month level (July 2016) in the filter expression in order to generate an analytically useful time series chart.

and infer the conjunctive expression, *e.g.*, "by Country sort Countries in descending order by average Sales." Even though VizQL requires *either* a group or an aggregation expression to generate a visualization, our system infers *both* a group and aggregation expression to address (*ii*). Figure 4a shows a group expression derived from the sort expression's attribute along with an aggregation expression generating a bar chart.

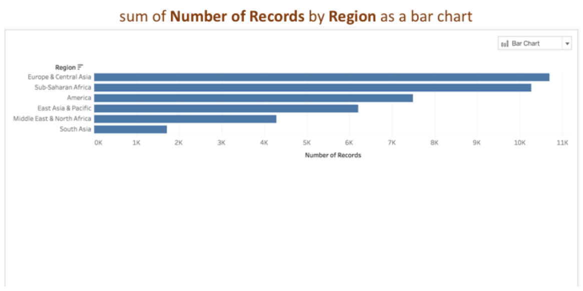
With respect to (*ii*), when a user types "temperature throughout july 2018", he likely expects the result to reveal the temporal aspects of the data. ArkLang supports the notion of level of detail in data hierarchies such as location and time. To generate a time-series line chart, the system introspects the current level of detail of a temporal hierarchy in the filter expression, and infers a group expression of the temporal concept to be 1 level lower than the original temporal concept in the filter expression (except for 'second', which is the lowest level of the temporal hierarchy; in this case, the system simply infers 'second'). Figure 4b shows the result of this inferencing logic.

5.4 Inferencing for supporting visualization types

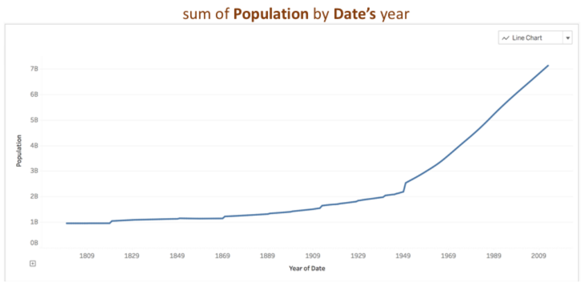
During visual analysis, people may *explicitly* express their intent for a specific graphical representation, such as a line chart to perform temporal analysis. The inferencing logic for deducing sensible attributes to satisfy valid visualizations, relies on *Show Me*, an integrated set of rules and defaults, incorporating automatic presentation from the row and column structure of a VizQL expression [21]. Show Me adopts best practices from graphics design and information visualization when ranking analytically useful visualizations based on the type of attributes utilized in the analysis workflow.

Amongst the visualizations we support in our system, text tables have the lowest rank because their primary utility is to look up specific values. The higher ranked visualizations present views that encode data graphically. Since text tables have a low rank, their condition is easily met and is always available as a default visualization. Hence, we do not have to infer any attributes to display a text table. We support 7 other visualizations and enumerate their corresponding inferencing logic when a user explicitly asks for these chart types in their input utterances:

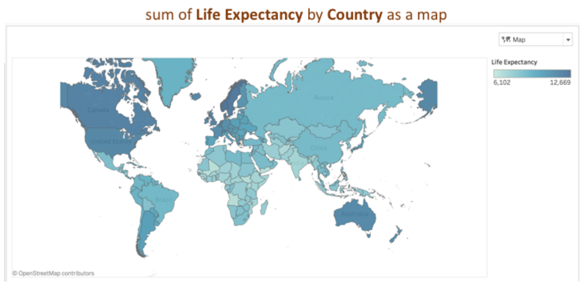
- **Bar chart**: Infer a quantitative attribute, as bars are effective for comparing numerical values, particularly when they are aligned. *E.g* In "start date as a bar chart," infer 'sum of Number of Records' to return a result "by Start Date's year and sum of Number of Records as a bar chart."
- **Gantt chart**: Gantt charts are effective for showing duration for a quantitative attribute. Infer a date attribute when only a dimension is present; infer a dimension when only a date attribute is present; infer both a dimension and a date time field if both are not present. *E.g.* In "order date as a gantt," infer 'Category' to return "sum of Number of Records by Order Date's year and by Category as a gantt chart."
- **Line chart**: A line chart is effective for showing trends. This command treats the date field discretely. Infer a date attribute. *E.g.* In "sum of sales by customer name as a line chart," infer 'Order Date' to return a result "sum of Sales by Customer Name by Order Date's year as a line chart."
- **Map**: Infer a geographic attribute. *E.g.* In "sum of sales by customer name by location," infer 'City' to return a result "sum of Sales by Customer Name by City as a map."
- **Pie chart**: Pie charts are generally used to show percentage or proportional data represented by each category. Given a numerical attribute, infer a categorical attribute. *E.g.* In "sum of sales as a pie chart" infer 'Category' to return a result "sum of Sales by Category as a pie chart." Similarly, given a categorical attribute, infer a numerical one.
- **Scatter plot**: Scatter plots are effective for comparing two values. Infer a additional measure. *E.g.* In "correlate sum of sales by customer name," infer 'Discount' to return a result "sum of Sales and sum of Discount by Customer Name as a scatterplot."
- **Treemap**: Treemaps are used to display hierarchical data using nested rectangle representation. Given a numerical attribute, infer a dimension. *E.g.* In "sum of sales as a tree map" infer 'Category' to return a result "sum of Sales by Category as a treemap." Similarly, given a categorical attribute, infer a numerical one.



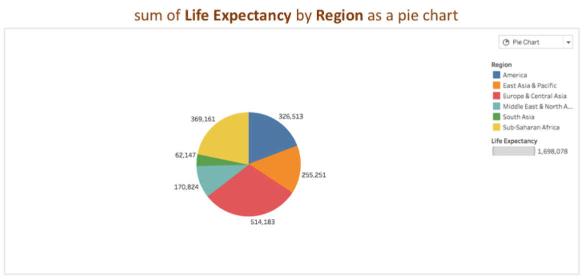
(a) "show me regions as a bar chart"



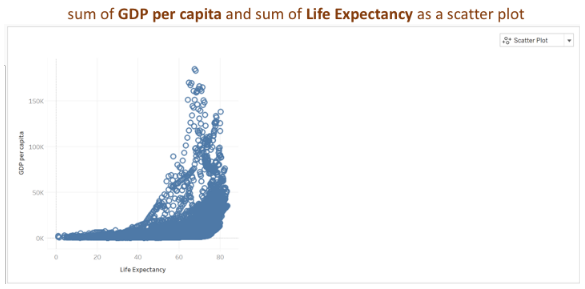
(b) "population over time"


sum of Life Expectancy by Country as a map
Map
Life Expectancy
6,102
12,669
© OpenStreetMap contributors

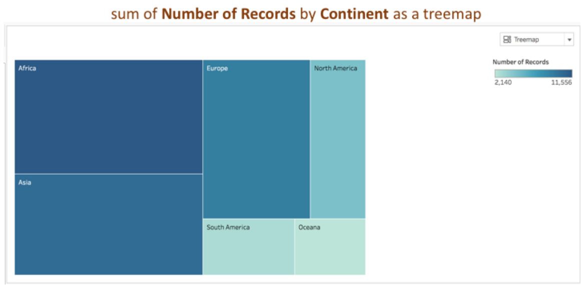
(c) "life expectancy by location"



(d) "show me life expectancy in a pie chart"



(e) "what's the correlation of gdp?"



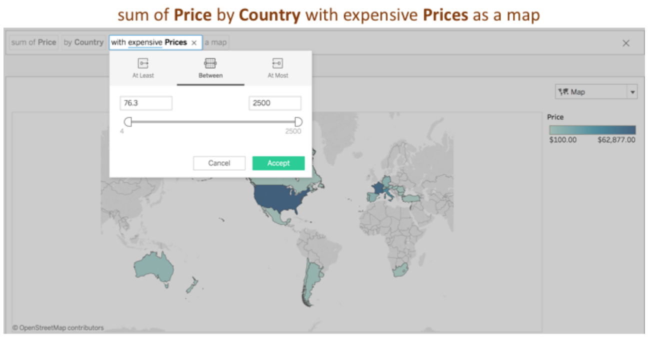
(f) "continent in a treemap"

Figure 5: Inferring attributes to support various visualization types expressed in the input utterance (a) A measure 'sum of Number of Records' is inferred to show the number of records per region, (b) A date field is inferred to generate a time series chart, (c) A geographic field 'Country' is inferred, (d) A categorical attribute is inferred to generate a pie chart, (e) A top-ranked measure based on usage popularity is inferred to generate a scatter plot to show correlation, (f) A measure 'sum of Number of Records' is inferred to show the relative sizes of the continents.

One of the goals of our system is to provide analytically useful visualization responses to the user's data inquiry. We draw from best practices and principles in information visualization to translate such inferences into transformations on the visualization [29, 43]. For example, the utterance "top 3 employees with highest profit margins" generates a bar chart. In order to better help the user analyze the employee data quantitatively for comparison and ranking, we sort the bars in the bar chart in descending order as an *implicit* inference (as seen in Figures 2c, 2d and 3b). Similarly, when a user asks the question "house prices in seattle over the past 5 years," displaying geographic and temporal data in a time series line chart is often more effective than the same in a map.

5.5 Resolving vague predicates

Vagueness is a term used in linguistics manifested by concepts such as 'low', 'high', 'good', and 'near.' These concepts are termed as 'vague' because of our inability to precisely determine and generalize the extensions of such concepts in certain domains and contexts [18, 35]. Vague concepts typically have blurred boundaries without a clear distinction between the entities that fall within their extension and those that do not. With metadata provided by the Semantic Model, along with *Arklang*'s formalism of the various analytical expression types, we support various vague concepts. In addition, we extend our inferencing logic to make smart defaults for such concepts. For example, the vague concept *expensive* is a *filter expression*, whose syntactic and semantic structure expects an *attribute* that is of a currency type as encoded in the underlying



(a) “where are the expensive wines?”

Figure 6: Inferring vague predicates. The system infers a currency attribute ‘Price’ for the concept *expensive* and the geographic attribute ‘Country’ in response to the word ‘where’. A numeric range is inferred from the metadata for ‘Price.’ Clicking on ‘expensive’ enables refinement of selected values.

Semantic Model. The *filter* is *between* and the *value* is inferred by the attribute’s metadata also encoded in the Semantic Model. So, for the utterance “where are the expensive wines?”, the system infers *expensive* to range from $avg\ 1SD, max$, where avg, SD and max are the average, standard deviation and maximum values for the numerical field ‘Price’ that also has metadata indicating that it is a currency attribute. Similar to other systems such as Datatone and Eviza [12, 34], we expose these system presumptions as widgets where the user can override or redefine these defaults as shown in Figure 6. We collect telemetry data on these system overrides and interactions that provide a feedback loop to the system regarding relevancy in the inference logic.

6 DEPLOYMENT AND ITERATION

We deployed our system for a two month period, both within and outside our organization. Based on usage logs, at least 205 unique users tried the system, generating over 6200 visualizations. We observed considerable repeat use, with 66 people using the system at least 10 times each, and 10 people using it at least 50 times. A wide variety of sample datasources were available and people could also upload and use their own data.

We gathered feedback throughout the deployment and iteratively improved our inferencing logic to address concerns raised by our users. The logic evolved considerably over this time, as real world queries on various data sets identified limitations and gaps in our initial logic. The inferencing logic described in this paper represents our final set of rules following this iterative development period.

7 SUMMATIVE EVALUATION

As a summative assessment of our inferencing logic, we conducted a survey. Respondents assessed visualization response quality for a diverse set of underspecified input queries. To select input queries that were representative of actual use, we drew on collected log data from our two month deployment.

7.1 Method

7.1.1 Survey Data. We first curated a set of example queries and responses. From all logged input queries (459) on our most popular data source (a data set about wines), we manually filtered out intermediate log entries, nonsensical queries (e.g. “I love Pinot Noir”), queries that were unanswerable with the data source, fully specified inputs (where no inferencing is required), exact duplicate queries, and duplicate concepts. The result was a set of 35 queries covering all of the analytical concepts (but not equally represented) and most of the inferencing types.

For each query, we then used our system to generate a visualization. One complication is that our system generates several interpretations of any input utterance, from which the user may choose, rather than a single option. Our goal was to assess the quality of the ‘best’ of these possible interpretations, under the assumption that a user will know which one to choose. As such, for each query, two authors manually selected the best interpretation from the available list based on consensus agreement. We then used the system to generate a visualization for each of these ‘best’ options. Queries, visualizations, and survey results may be found in the supplemental material.

7.1.2 Procedure. Our survey asked respondents to assess the quality of each visualization in relation to its input query. Participants assessed both the system-selected *data content* (e.g., attributes, filters, aggregations) and the *visualization* (chart type / visual encoding), using two 5 point Likert scales ranging from <1 - very good> to <5 - very poor>. Prior to answering these questions, participants reviewed a detailed data schema to familiarize themselves with the data fields and their meaning.

Each participant rated all 35 visualizations. Questions were presented in random order. An optional comments box was provided for each question and participants were encouraged to add comments if they gave a rating of *poor* or *very poor*. Participants also answered several demographic questions.

7.1.3 Participants. Fifty-one adults from our organization (35 male, 15 female, 1 gender unspecified) completed the survey. Age varied between 20-65+ and job roles included software engineering, sales, research, and design. 21 of the participants had previously tried our natural language system during the deployment and 30 had not. Most participants regularly used data analytics tools, with 39 participants reporting daily or weekly use.

7.2 Results

Overall ratings of both data content and visualization design were quite positive, with 83% of all ratings being *Acceptable* or better (see Figure 7). Ratings of the data content and visualization were closely correlated.

Most questions received a wide range of ratings, but some had a much lower score distribution than others. Examining the lowest scoring instances revealed that they were mostly *dependent* sort or filter phrases that specify a sort or filter but not the desired visualization (e.g., “points > 90,” “sort by points,” “in the last 5 years”). Figure 8 illustrates the ratings difference between questions containing only dependent phrases (sort, filter, or limit) versus all others (N/A).

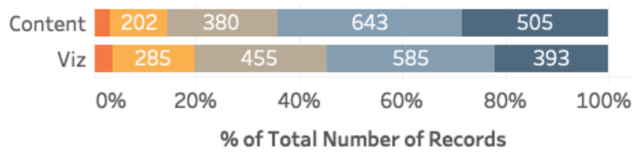


Figure 7: Overall ratings for data content and visualization (viz) design. Dark orange = very poor, beige = acceptable, dark blue = very good.

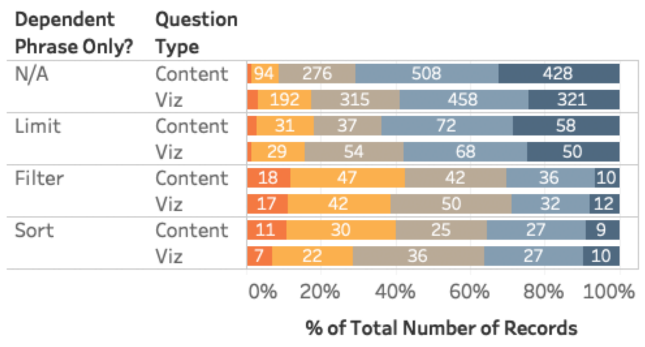


Figure 8: Ratings for questions that included only a dependent phrase (limit, filter, or sort) versus all other questions. Dark orange = very poor, beige = acceptable, dark blue = very good.

Questions containing only a limit expression received higher ratings than those with only a sort or filter, likely because attributes in a limit expression can be re-used within inferred expressions. For example, "5 most expensive titles" is interpreted by our system as "most expensive Price, by Title, top 5 Titles by most expensive Price". In contrast, filter and sort expressions give no hint about the desired visualization. In response to the query "sort by points", our system had to infer a dimension as a grouping variable, to create a list of items that could subsequently be sorted by points. With no knowledge about the meaning of dimensions, it chose *Description* (a paragraph of text) rather than the more useful *Title* for grouping. This prompted comments such as, *"Description is the last thing I'd expect,"* reflecting an expectation that the system would have semantic knowledge about attribute importance, or at least the key attribute characterizing each record.

Apart from dependent phrases, questions that required inferring only aggregation or visualization types had higher score distributions than those that required inferring data fields. One poorly rated question was "top 5 wines by variety," which our system interpreted as "sum of Number of Records by Variety and by Title, top 5 Titles by sum of Number of Records." Participants identified two inferencing problems: 1) they expected 'top' to be interpreted as the quality rating 'Points' rather than the default Number of Records, and 2) they expected the top 5 for each variety, not the top 5 sorted by variety. As one participant said, *"The real issue is that 'top' for me would be...either price or rating...however, this is difficult to infer as it would be domain specific...."*

8 DISCUSSION

Results of our summative evaluation are promising, with most examples receiving positive ratings. Our survey generated rapid feedback on a variety of inferencing cases, but we note several limitations. Participant responses to inferred visualizations might differ in a real analysis situation where they entered their own queries, completed a real analysis task, and could clarify ambiguities. Additionally, the survey questions were not representative, as we intentionally included only underspecified queries. Some of our examples may be more underspecified than we would expect in real situations, especially the bare filter and sort phrases that were problematic for our inferencing algorithm. Furthermore, the survey included only 35 queries on one dataset; future evaluations should consider more diverse data sources and questions.

Our paper focused on providing reasonable defaults for inferring underspecified analytical utterances. The survey helps us understand how inferencing of underspecified analytical utterances fares, and also suggests some promising areas for future work, where improved inferencing logic could be helpful. We find that users' explicit judgments for the same utterances can widely differ. Future improvements can focus on personalizing the inferencing logic by discerning individuals' unique analytical goals through relevancy judgments. Such judgments can be implicit through usage data or explicit through repair and refinement. We could analyze this gap by providing support to help users articulate their needs (*e.g.*, soliciting greater elaboration about a user's intent, providing suggestions based on the data or previous search behavior, or grouping similar interpretations for utterances) [42].

Our system often defaults to inferring the calculated measure 'Number Of Records', which is a reasonable default if the measure is not known, but was not wildly popular in the survey. We could improve the choice of which measure to infer based on personalization or additional semantics from the data itself [22]. For example, when a user asks about her patients in her healthcare organization, the number of visits or check-ins might be a more semantically appropriate measure. By leveraging context and semantics derived from entity relationships in the data, we may be able to infer more appropriate attributes. Measure popularity may also be useful here.

Interestingness of data patterns is another promising area of research of natural language systems supporting visual analysis. Data interestingness emphasizes a notion of diversity, novelty, surprisingness, such as outliers and probability distributions deviating from the uniform distribution [13]. While several metrics from statistics and information retrieval have been proposed to measure interestingness [41], it would be interesting to explore such metrics when inferring attributes in an utterance beyond just the data types.

9 CONCLUSION

Natural language interfaces are becoming a useful modality for exploring data and garnering insights. However, terse and underspecified input, the bane of traditional web search systems, is a challenge. In this paper, we introduce a set of inferencing techniques to help generate useful visualization responses to underspecified utterances. Based on constraints of the underlying analytics platform and imbibing best practices from information visualization literature, we support four types of inferencing - handling underspecification within the analytical expressions, handling underspecification between the analytical expressions, inferring attributes given an explicit intent for a visualization type, and inferring reasonable defaults for vague underspecified concepts. Our inferencing heuristics were iteratively refined over a two month field trial based on feedback from users. With the exception of bare filter and sort expressions that do not define the intended visualization content, a summative evaluation survey revealed an overall positive response to our inferencing approach. In summary, our findings pave the way for a promising area of interdisciplinary research, drawing inspiration from web search, data mining, visual analytics, and personalization.

REFERENCES

[1] 2002. Placing Search in Context: The Concept Revisited. *ACM Trans. Inf. Syst.* 20, 1 (Jan. 2002), 116–131. DOI:http://dx.doi.org/10.1145/503104.503110

[2] Adnan Abid, Naveed Hussain, Kamran Abid, Farooq Ahmad, Muhammad Shoaib Farooq, Uzma Farooq, Sher Afzal Khan, Yaser Daanial Khan, Muhammad Azhar Naeem, and Nabeel Sabir. 2016. A survey on search results diversification techniques. *Neural Computing and Applications* 27, 5 (2016), 1207–1229. DOI:http://dx.doi.org/10.1007/s00521-015-1945-5

[3] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. 2009. Diversifying Search Results. In *Proc. 2nd ACM Intl. Conf. Web Search and Data Mining (WSDM '09)*. ACM, New York, NY, USA, 5–14. DOI:http://dx.doi.org/10.1145/1498759.1498766

[4] Ion Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. 1995. Natural language interfaces to databases–an introduction. *Natural language engineering* 1, 1 (1995), 29–81. DOI:http://dx.doi.org/10.1017/S135132490000005X

[5] Johan Bos. 2004. Computational Semantics in Discourse: Underspecification, Resolution, and Inference. *J. Logic, Language and Information* 13, 2 (Mar 2004), 139–157. DOI:http://dx.doi.org/10.1023/B:JLLI.0000024731.26883.86

[6] Jay Budzik and Kristian J. Hammond. 2000. User Interactions with Everyday Applications As Context for Just-in-time Information Access. In *Proc. 5th Intl Conf. Intelligent User Interfaces (IUI '00)*. ACM, New York, NY, USA, 44–51. DOI:http://dx.doi.org/10.1145/325737.325776

[7] Gabriele Capannini, Franco Maria Nardini, Raffaele Perego, and Fabrizio Silvestri. 2011. Efficient Diversification of Web Search Results. *Proc. VLDB Endow.* 4, 7 (April 2011), 451–459. DOI:http://dx.doi.org/10.14778/1988776.1988781

[8] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman. 1999. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 579–581 pages. http://dl.acm.org/citation.cfm?id=300679.300826

[9] Noam Chomsky. 1957. *Syntactic Structures*. Mouton and Co., The Hague.

[10] Kenneth Church and Ramesh Patil. 1982. Coping with Syntactic Ambiguity or How to Put the Block in the Box on the Table. *Comput. Linguist.* 8, 3-4 (July 1982), 139–149. http://dl.acm.org/citation.cfm?id=972942.972946

[11] Kedar Dhamdhere, Kevin S. McCurley, Ralfi Nahmias, Mukund Sundararajan, and Qiqi Yan. 2017. Analyza: Exploring Data with Conversation. In *Proc. 22nd Intl Conf. on Intelligent User Interfaces (IUI '17)*. ACM, New York, NY, USA, 493–504. DOI:http://dx.doi.org/10.1145/3025171.3025227

[12] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G. Karahalios. 2015. DataTone: Managing Ambiguity in Natural Language Interfaces for Data Visualization. In *Proc. ACM Symp. User Interface Software Technology (UIST 2015)*. ACM, New York, NY, USA, 489–500. DOI:http://dx.doi.org/10.1145/2807442.2807478

[13] Liqiang Geng and Howard J. Hamilton. 2006. Interestingness Measures for Data Mining: A Survey. *ACM Comput. Surv.* 38, 3, Article 9 (Sept. 2006). DOI:http://dx.doi.org/10.1145/1132960.1132963

[14] Lars Grammel, Melanie Tory, and Margaret-Anne Storey. 2010. How information visualization novices construct visualizations. *IEEE Trans. visualization & computer graphics* 16, 6 (2010), 943–952. DOI:http://dx.doi.org/10.1109/TVCG.2010.164

[15] Barbara J. Grosz, Douglas E. Appelt, Paul A. Martin, and Fernando C. N. Pereira. 1987. TEAM: An Experiment in the Design of Transportable Natural-language Interfaces. *Artif. Intell.* 32, 2 (May 1987), 173–243. DOI:http://dx.doi.org/10.1016/0004-3702(87)90011-7

[16] Sumit Gulwani and Mark Marron. 2014. NLyze: Interactive Programming by Natural Language for Spreadsheet Data Analysis and Manipulation. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14)*. ACM, New York, NY, USA, 803–814. DOI:http://dx.doi.org/10.1145/2588555.2612177

[17] E. Hoque, V. Setlur, M. Tory, and I. Dykeman. 2018. Applying Pragmatics Principles for Interaction with Visual Analytics. *IEEE Trans. Visualization & Computer Graphics* 24, 1 (Jan. 2018), 309–318. DOI:http://dx.doi.org/10.1109/TVCG.2017.2744684

[18] Dominic G. Hyde. 2010. Vagueness, Logic and Ontology. *Bulletin of Symbolic Logic* 16, 4 (2010), 531–533.

[19] Tadao Kasami. 1966. An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257* (1966).

[20] Fei Li and Hosagrahar V Jagadish. 2014. NaLIR: An interactive natural language interface for querying relational databases. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (06 2014). DOI:http://dx.doi.org/10.1145/2588555.2594519

[21] J. Mackinlay, P. Hanrahan, and C. Stolte. 2007. Show Me: Automatic Presentation for Visual Analysis. *IEEE Trans. Visualization & Computer Graphics* 13, 6 (Nov 2007), 1137–1144. DOI:http://dx.doi.org/10.1109/TVCG.2007.70594

[22] Stuart Madnick and Hongwei Zhu. 2006. Improving Data Quality Through Effective Use of Data Semantics. *Data & Knowledge Engineering* 59, 2 (Nov. 2006), 460–475. DOI:http://dx.doi.org/10.1016/j.datak.2005.10.001

[23] James Mayfield, Tim Finin, and others. 2003. Information retrieval on the Semantic Web: Integrating inference and retrieval. In *Proc. SIGIR Workshop on the Semantic Web*.

[24] Lilyana Mihalkova and Raymond Mooney. 2009. Learning to Disambiguate Search Queries from Short Sessions. In *Proc. European Conf. Machine Learning and Knowledge Discovery in Databases: Part II (ECML PKDD '09)*. Springer-Verlag, Berlin, Heidelberg, 111–127. DOI:http://dx.doi.org/10.1007/978-3-642-04174-7_8

[25] John F. Pane and Brad A. Myers. 2001. Studying the language and structure in non-programmers' solutions to programming problems. *Intl J.Human-Computer Studies* 54, 2 (2001), 237–264. DOI:http://dx.doi.org/10.1006/ijhc.2000.0410

[26] Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a Theory of Natural Language Interfaces to Databases. In *Proceedings of the 8th International Conference on Intelligent User Interfaces (IUI '03)*. ACM, New York, NY, USA, 149–157. DOI:http://dx.doi.org/10.1145/604045.604070

[27] Feng Qiu and Junghoo Cho. 2006. Automatic Identification of User Interest for Personalized Search. In *Proc. 15th Intl Conf. World Wide Web (WWW '06)*. ACM, New York, NY, USA, 727–736. DOI:http://dx.doi.org/10.1145/1135777.1135883

[28] Daniel E. Rose and Danny Levinson. 2004. Understanding User Goals in Web Search. In *Proc. 13th Intl Conf. World Wide Web (WWW '04)*. ACM, New York, NY, USA, 13–19. DOI:http://dx.doi.org/10.1145/988672.988675

[29] H. Rosling, A.R. Rönnlund, and O. Rosling. 2018. *Factfulness: Ten Reasons We're Wrong About the World–and Why Things Are Better Than You Think*. Flatiron Books. https://books.google.com/books?id=j-4yDwAAQBAJ

[30] Hassan Sajjad, Patrick Pantel, and Michael Gamon. 2012. Underspecified Query Refinement via Natural Language Question Generation. In *COLING 2012, 24th Intl Conf. Computational Linguistics*. Mumbai, India, 2341–2356. http://aclweb.org/anthology/C/C12/C12-1143.pdf

[31] Gerard Salton and Michael J. McGill. 1986. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA.

[32] Rodrygo L.T. Santos, Craig Macdonald, and Iadh Ounis. 2010. Exploiting Query Reformulations for Web Search Result Diversification. In *Proc. 19th Intl Conf. World Wide Web (WWW '10)*. ACM, New York, NY, USA, 881–890. DOI:http://dx.doi.org/10.1145/1772690.1772780

[33] Emanuel A. Schegloff, Gail Jefferson, and Harvey Sacks. 1977. The Preference for Self-Correction in the Organization of Repair in Conversation. *Language* 53, 2 (1977), 361–382. http://www.jstor.org/stable/413107

[34] Vidya Setlur, Sarah E. Battersby, Melanie Tory, Rich Gossweiler, and Angel X. Chang. 2016. Eviza: A Natural Language Interface for Visual Analysis. In *Proc. Symp. User Interface Software and Technology (UIST 2016)*. ACM, New York, NY, USA, 365–377. DOI:http://dx.doi.org/10.1145/2984511.2984588

[35] Stewart Shapiro. 2006. *Vagueness in Context*. Oxford University Press.

[36] Ben Shneiderman. 1997. Direct Manipulation for Comprehensible, Predictable and Controllable User Interfaces. In *Proc.2nd Intl Conf. Intelligent User Interfaces (IUI '97)*. ACM, New York, NY, USA, 33–39. DOI:http://dx.doi.org/10.1145/238218.238281

[37] David Sontag, Kevyn Collins-Thompson, Paul N. Bennett, Ryen W. White, Susan Dumais, and Bodo Billerbeck. 2012. Probabilistic Models for Personalizing

Web Search. In *Proc. 5th ACM Intl Conf. Web Search and Data Mining (WSDM '12)*. ACM, New York, NY, USA, 433–442. DOI:**http://dx.doi.org/10.1145/2124295.2124348**

[38] Arjun Srinivasan and John Stasko. 2018. Orko: Facilitating Multimodal Interaction for Visual Exploration and Analysis of Networks. *IEEE Trans. Visualization & Computer Graphics* 24, 1 (2018), 511–521. DOI:**http://dx.doi.org/10.1109/TVCG.2017.2745219**

[39] Sofia Stamou and Alexandros Ntoulas. 2009. Search Personalization Through Query and Page Topical Analysis. *User Modeling and User-Adapted Interaction* 19, 1-2 (Feb. 2009), 5–33. DOI:**http://dx.doi.org/10.1007/s11257-008-9056-y**

[40] Chris Stolte, Diane Tang, and Pat Hanrahan. 2002. Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. *IEEE Trans. Visualization & Computer Graphics* 8, 1 (Jan. 2002), 52–65. DOI:**http://dx.doi.org/10.1109/2945.981851**

[41] Pang-Ning Tan and Vipin Kumar. 2000. Interestingness measures for association patterns: A perspective. In *Proc. Workshop on Postprocessing in Machine Learning and Data Mining*.

[42] Jaime Teevan, Susan T. Dumais, and Eric Horvitz. 2010. Potential for Personalization. *ACM Trans. Comput.-Hum. Interact.* 17, 1, Article 4 (April 2010), 31 pages. DOI:**http://dx.doi.org/10.1145/1721831.1721835**

[43] Edward R. Tufte. 1986. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, USA.

[44] Michael J. Welch, Junghoo Cho, and Christopher Olston. 2011. Search Result Diversity for Informational Queries. In *Proc. 20th Intl Conf. World Wide Web (WWW '11)*. ACM, New York, NY, USA, 237–246. DOI:**http://dx.doi.org/10.1145/1963405.1963441**