Talk2Data: High-Level Question Decomposition for Data-Oriented Question and Answering

Danqing Shi, Yi Guo, Mingjuan Guo, Yanqiu Wu, Qing Chen and Nan Cao

Abstract— Through a data-oriented question and answering system, users can directly “ask” the system for the answers to their analytical questions about the input tabular data. This process greatly improves user experience and lowers the technical barriers of data analysis. Existing techniques focus on providing a concrete query for users or untangling the ambiguities in a specific question so that the system could better understand questions and provide more correct and precise answers. However, when users have little knowledge about the data, it is difficult for them to ask concrete questions. Instead, high-level questions are frequently asked, which cannot be easily solved with the existing techniques. To address the issue, in this paper, we introduce Talk2Data, a data-oriented online question and answering system that supports answering both low-level and high-level questions. It leverages a novel deep-learning model to resolve high-level questions into a series of low-level questions that can be answered by data facts. These low-level questions could be used to gradually elaborate the users’ requirements. We design a set of annotated and captioned visualizations to represent the answers in a form that supports interpretation and narration. We evaluate the effectiveness of the Talk2Data system via a series of evaluations including case studies, performance validation, and a controlled user study. The results show the power of the system.

Index Terms—Natural Language Interfaces;

1 INTRODUCTION

The data-oriented question and answering is a process in which users ask natural language questions about the input data and the system extracts relevant data facts [50,57] and presents them in visual forms to answer the question. Recent techniques and systems in this topic attract great attention in both academics [6,20,32,33,35,49,53] and industry [1,3]. These systems provide a more intuitive communication approach and better user experience for data analysis. Users can directly “tell” the analysis system their needs or “ask” the system for the answers to their questions about the input data. This process greatly lowers the barriers of data analysis, but the correctness of the response highly depends on the system’s capability of understanding users’ questions that are presented in natural language.

To improve the accuracy, existing techniques are designed either to guide users to provide a more concrete nature language query [55] or untangle ambiguities in the input query [20] to better capture user requirements and more precisely drive the underlying data analysis for question answering. Following these ideas, most recently, Arpit et al. introduce NL4DV [35], an open source python toolkit that integrates many state-of-the-art techniques [20,32,33] to translate user queries into a high-level visualization grammar [46]. The toolkit is able to map precise queries with concrete requirements to low-level analysis tasks, which significantly lowers the technique barriers of building a nature language interface (NLI) for visualization. However, when users have little knowledge about the data, it is almost impossible for them to ask precise and concrete questions. High-level questions such as “what causes global warming” are frequently asked, which could be decomposed into multiple low-level tasks and cannot be easily resolved.

Fig. 1. A case study in which a user asks the Talk2Data system two high-level questions (b) to explore a marketing dataset about car sales (a). Two high-level questions are decomposed into several low-level questions and answered by a set of annotated charts (c).

- Danqing Shi, Yi Guo, Mingjuan Guo, Yanqiu Wu, Qing Chen and Nan Cao are with Intelligent Big Data Visualization Lab, Tongji University. E-mail: {sdq, 2010937, guomingjuan, 1941923, qingchen, nancao} @tongji.edu.cn. Nan Cao is the corresponding author.

Manuscript received xx xxx, 201x; accepted xx xxx, 201x. Date of Publication xx xxx, 201x; date of current version xx xxx, 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx
by the existing techniques [52]. To resolve such a high-level question is difficult. Many challenges exist: first, high-level questions usually cover multiple data dimensions and correspond to multiple low-level analysis tasks that are difficult for a system to differentiate. Second, high-level questions usually cannot be directly answered without mentioning the context and elaborating the details in multiple different aspects. It is difficult to correctly extract these contextual information and details from the data merely based on a fuzzy question. Third, to better present the answers, the extracted contextual information and data details should be organized in order and visualized in a form that facilitates result narration and interpretation.

To address the above challenges, we introduce Talk2Data, a data-oriented question and answering system that supports natural language queries about an input spreadsheet given by both low-level (specific) and high-level (fuzzy) questions. In particular, the system first employs a novel deep-learning based question decomposition model to resolve a high-level question into a series of relevant low-level questions. After that, a search algorithm is introduced to explore the data space and extract meaningful facts that are most relevant to each low-level question. These facts are finally shown in the visualizations as the parts of the various systems have been explored both within the research community of NL interpretation, existing NLIs are designed either to guide the users to provide a more concrete nature language query [15, 49, 65] or untangle ambiguities in the input query [3][20][53][58] to better capture user requirements and more precisely drive the underlying data analysis for question answering.

The initial prototype of visualization NLI [15] does not rely on any intelligent approach to interpret user questions but create a set of supporting commands to guide users’ inputs. Flowsense [65] and Eviza [20] depends on the pre-defined grammar to capture query patterns. When the user types a partial query and pauses, the system triggers the feature of query auto-completion to guide users’ queries. However, the grammar-based methods limits the range of questions as it is impossible to cover all of the possible tasks.

To improve the flexibility in posing data-related questions while managing ambiguities in NL queries, many NLIs leverage the sophisticated NLP parsing techniques (e.g., dependencies) to understand the intuitions of queries and detect ambiguities present in the queries. In Articulate [53], the translation of user’s imprecise specification is based on a NL parser imbued with machine learning algorithms that are able to make reasoned decisions automatically. DataTone [20] adopts a mixed-initiative approach to manage ambiguities in NLIs. Specifically, the system displays ambiguity widgets along with the main visualization, therefore users are allowed to switch the content in widgets to get desired alternative views. The idea in DataTone is extended to NL4DV toolkit [55], a python-based library released to translate user queries into a high-level visualization grammar. For the developers without experience with NLP, NL4DV can save their efforts on learning NLP knowledge when building the visualization NLIs.

The aforementioned NLIs are only able to answer the precise low-level questions. The grammatical-based methods do not support high-level questions as interpreting multiple tasks in one query poses parsing difficulty [65]. On the other hand, the NLP parsing techniques leverage rule-based parsers to comprehend user instructions and questions. Thus, when a question does not specifically contain keywords for analytic tasks, these interfaces cannot precisely understand user intentions. However, for the users with few knowledge about the data, it is almost impossible for them to ask precise and concrete question. In practically, the usability of these systems is under user expectations. In order to overcome these limitations, we build a novel deep-learning based model that not only can resolve a high-level (fuzzy) question into a series of relevant low-level questions (specific), but also improve the robustness of NL interpretation. In Talk2Data, users can ask both low-level and high-level questions about a table, and get well-designed diagrams that represent the facts extracted for answering the input question.

2.2 Question Answering System

Question Answering (QA) is a well-researched area about building systems that can answer NL questions. Advances in NLP facilitate the development of various QA systems, such as Text-Based QA [17, 19, 25, 34, 42, 56], Knowledge-Based QA [9][11][47], and Table-Based QA [26, 40, 63, 64].

In this paper, our work concerns about Table-Based QA, where we are tasked to answer both high-level and low-level queries given a table. The existing Table-Based QA system, such as [16, 40, 63], are designed to answer the low-level questions. Given a NL query and a table, Neural Enquirer [63] first encodes the query and table into distributed representations, and then use a multi-layer executor to derive the answer. It can be trained using Query-Answer pairs, where the distributed representations of queries and the table are optimized together with the query execution logic in an end-to-end fashion. However, Cho et al. [14] stated that only using the answer annotated dataset for training and evaluation may count “spurious” programs, the system will accidentally lead to correct answers by using the wrong cells or operations. Therefore, the authors propose a multi-layer sequential network with attention supervision to answer questions; it uses multiple Selective Recurrent Units to improve the interpretability of the model. Moreover, translating NL to SQL queries is a commonly used approach to answer the questions related to spreadsheets or database. Seq2SQL [65] leverages the policy-based reinforcement learning to translate the NL queries to corresponding SQL queries. SQLNET [61] proposes a sequence-to-set model and a column attention mechanism

© 2022 Talk2Data GitHub
to synthesize the SQL queries from NL as a slot filling problem. In the past few years, the large-scale pre-trained language models have rapidly improved the ability to understand and answer the free-form questions. The most recent work, TaBERT [64], is a pre-trained language model that is built on top of BERT [16] to jointly learn contextual representations for queries and tables, which can be used as encoder of query and table in QA systems.

Although efficient, the above existing Table-Based QA systems are designed to conduct information retrieval tasks, which cannot answer the questions requiring to analyzing the data. Moreover, the input questions for existing systems have to be precise and concrete. When the input question contains multiple tasks or drops the anchor words, the accuracy of system will be strongly impacted [40]. Existing works, proposed to resolving high-level questions, are concentrated on text-based QA, such as [14] [42]. To our best knowledge, there is no prior study working on resolving the high-level questions about a table. To fill this gap, we introduce Talk2Data, a data-oriented question and answering system that supports both low-level and high-level questions. In order to answer the high-level questions, in Talk2Data, we adopt a novel decomposition model to resolve the high-level questions into a series of low-levels that can be answered by data facts. The decomposition model extends the classic sequence to sequences architecture [55] and integrate attention and copy mechanism to guide the generation of each low-level questions.

### 2.3 Question Answering Corpus

Various QA corpus, such as text-based QA corpus [5] [22] [23] [27] [28] [43] [55] [62], knowledge-based QA corpus [9] [11] [12] [55], and table-based QA corpus [13] [40], have been constructed to train the deep learning models for QA with the goal of improving the accuracy.

Our corpus is designed for decomposing high-level questions into a series of relevant low-level questions. The works that are most relevant to ours are table-based QA corpus [13] [40] [67] and question decomposition corpus [59]. WIKITABLEQUESTIONS [40], consisting of 22k question-answer pairs, is the most frequently used corpus in table-based QA, followed by Break [59] a large-scale Text-to-SQL dataset which consists of 10k questions, and 5k corresponding complex SQL queries. Break [59] is a question decomposition dataset that contains human composed questions sampled from other QA benchmarks, it was aimed at training models to reason over complex questions. It collects over 80k NL questions, annotated with a new meaning representation and question decomposition meaning representation. Break is the most relevant corpus, but it does not contain any question relating to visual analysis. Therefore, in our work, we build the first question decomposition corpus in the domain of visual analysis, it consists of 9,071 high-level questions with each question corresponds to two low-level questions.

### 3 Overview and System Design

In this section, we describe the design requirements of the Talk2Data system, followed by an introduction of the system architecture and the problem formulation.

#### 3.1 Design Requirements

Our goal is to design and develop a data-oriented question and answering system that is able to automatically extract data facts from an input spreadsheet to answer users’ high-level questions about the data. To achieve the goal, a number of requirements should be fulfilled:

- **R1** Elaborate high-level questions in context. The system should be able to resolve high-level questions and elaborate the problem gradually from different aspects to give a comprehensive answer in context of the input data.
- **R2** Rank the answers. The system should be able to rank the potential answers, i.e., data facts, in order according to their relevance to the question.
- **R3** Clear answers narration. The answers to a high-level question, should be visualized with narrative information such as captions and annotations and arranged in a logic order so that the users can easily read and understand them in a short time.
- **R4** Real-time communication and responding. To improve the user experience, the system should be able to support real-time query and should search for the results and respond to users immediately without latency.

#### 3.2 System Architecture and Formulation

To fulfill the above requirements, as shown in Fig. 2 we design Talk2Data system with four major modules: (a) the preprocessing module, (b) the decomposition module, (c) the answer seeking module, and (d) the answer representation module. In particular, the preprocessing module parses the input tabular data and the corresponding question $Q$, and combines the parsing results together as word sequence $Q_{x}$ in the following form to facilitate computation:

$$Q_{x} \leftarrow [w_{1}, w_{2}, ..., w_{n}, \langle N \rangle, c_{n}, ..., \langle T \rangle, c_{t}, ..., \langle C \rangle, c_{c}, ...] \quad (1)$$

where $w_{i}$ is a word / phrase in $Q$ and $c_{i}$ is a column in $X$ and $\langle N \rangle \langle T \rangle \langle C \rangle$ shows its corresponding data type, i.e., numerical ($N$), temporal ($T$) and categorical ($C$) respectively.

The decomposition module introduces a deep learning model by extending the classic sequence-to-sequence model to resolve a high-level data-oriented question (represented by $Q_{x}$) into a series of relevant low-level questions that cover different aspects of the question to guide the answer seeking process (R1):

$$[q_{1}, q_{2}, ..., q_{m}] \leftarrow \text{Decompose}(Q_{x}) \quad (2)$$

where $s_{j}$ is a hidden vector corresponding to a low-level question that can be answered by specific data facts. The details about the decomposition algorithm is discussed in Section [4]
In the answer extraction module, the system searches the data space $X$ to extract data facts $f_i$ that are relevant to each of the low-level questions $q_i$ and rank them to find out the answers (R2). The whole process is based on a parallel beam-search algorithm that guarantees the performance requirement as described in (R4). This step can be formally presented as:

$$[f_1, f_2, \ldots, f_n] \leftarrow \text{Extract}(q, X) \quad (3)$$

where each data fact $f_i$ is a potential answer to the question $q_i$. The facts are ordered based on their relevance to the question. We define the data fact $f_i$ as a 5-tuple following the definition introduced in [50], which is briefly described as follows:

$$f_i = \{\text{type}, \text{subspace}, \text{breakdown}, \text{measure}, \text{focus}\}$$

where type (denoted as $t_i$) indicates the type of analysis task of the fact, whose value is one of the following cases: showing value, difference, proportion, trend, categorization, distribution, rank, association, extreme, and outlier; subspace (denoted as $s_i$) is the data scope given by a set of filters; breakdown (denote as $h_i$) is given by a temporal or categorical data fields based on which the data items in the subspace can be divided in groups; measure (denote as $m_i$) is a numerical data field based on which the program can retrieve a data value or compute a derived aggregated value in the subspace or each data group; focus (denote as $x_i$) indicates a set of specific data items in the subspace that require extra attention.

Finally, the representation module organizes the data facts in order and visualizes them via a set of captioned and annotated charts (R3) that are specifically designed to help with the narration of data semantics. In the following sections, we will describe the technique details of the decomposition, answer extraction, and representation modules.

4 QUESTION DECOMPOSITION

In this section, we first introduce the two basic question decomposition methods followed by a detailed description of the decomposition algorithm and model. After that, we introduce a data corpus that we collected to train our model. At last, we briefly describe how we implement our algorithm.

4.1 Question Types and Decomposition Strategies

Our system is designed to resolve two types of high-level questions: Type-I, the compound questions combining multiple low-level tasks such as “what aspects increase or decrease when the temperature raises?”; and Type-II, the fuzzy questions only describing potential scopes without mentioning any task such as “what causes global warming”? The first type can be resolved by enumerating and separating the low-level tasks mentioned in the question into individual low-level questions. In this way, the first example question can be resolved into two low-level questions “what aspects increase when the temperature raises?” and “what aspects decrease when the temperature raises?” The second type of high-level questions can be resolved by exploring and enumerating all possible analysis tasks and data scopes to formulate low-level questions. In this way the second example question about global warming can be resolved as a number of low-level questions such as “is the increasing of carbon dioxide associated with temperature raising” and “is the increasing of sunshine-time associated with temperature raising”, where association is a selected task and carbon dioxide and sunshine-time are the selected data columns that could potentially answer the question.

4.2 Decomposition Algorithm

Fig. 3 illustrates the running pipeline of the system’s question decomposition algorithm. Given an input question $Q$ the algorithm first check if the question is a high-level or low-level question based on a pre-trained classifier. The low-level questions will be directly send to the system’s answering module, but the high-level ones will be decomposed iteratively into a series of relevant low-level questions via a deep decomposition model. The model resolves an input high-level question $Q_h$ into a set of sub-questions $(q_1, q_2, \ldots, q_n)$. If $q_i$ is a low-level question, it will be directly answered, otherwise it will be decomposed again. This process runs iteratively until all the high-level questions are resolved.

4.3 Decomposition Model

Fig. 4. Schematic diagrams of the decomposition model. It consists of four major improvements: (a) question classifier, (b) decomposition layer, (c) attention mechanism, and (d) copying mechanism.

The decomposition model is designed by extending the classic sequence-to-sequence model [54] from four aspects: (1) we pre-train a BERT-classifier [15] to compute a conditional vector for each input question $Q_h$ to indicates the question type (i.e., Type-I or Type-II) so a proper decomposition method could be chosen by the model; (2) we add a decomposition layer in the model to transform the encoding vector $h$ of $Q_h$ into two hidden vectors $(h_{r1}, h_{r2})$ corresponding to two output questions; (3) we employ the attention mechanism [7, 31] in the model to ensure and enhance the relevance between the input high-level question and the output questions. (4) we integrate a copying mechanism [21] to generalize the model so that it could make a correct response when users ask questions about a new dataset beyond the training corpus. To make it simple, our model decompose a high-level question to exactly two sub-questions that could either be a low-level question or another high-level question.

Specifically, given the formulated question $Q_h$, an embedding layer $W_{emb}$ projects each of the input words into a latent vector $v$, which is further encoded by a bidirectional GRU encoder:

$$h = \text{encode}(v), \quad v = \text{embed}(Q_h) \quad (4)$$

where $h$ is the vector representation of $Q_h$ that captures the semantics of the input question and the corresponding tabular data.

At the same time, as shown in Fig 5-a, a condition vector $c$ that indicates the question type is calculated by classifying $Q_h$ based on BERT [15], a large scale pre-trained nature language model:

$$c = \text{softmax}(W_u u), \quad u = \text{BERT}(Q_h) \quad (5)$$

where $u$ is the intermediate vector encoded by BERT and $W_u$ is a parameter matrix to be trained. The output $c$ is a two-dimensional one-hot conditional vector that indicates the question type with $[0, 1]$ indicating Type-I questions and $[1, 0]$ indicating Type-II questions. With the vector, the model is able to select a proper approach to decompose the input question.

In the next, a decomposition layer is introduced in the model to transform $h = [h, c]$ into two hidden vectors $h_{r1}$ and $h_{r2}$ (Fig 5-b). It is implemented by a fully connected feed-forward neural network. Formally, the decomposition process is defined as follows:

$$h_{r1} = \tanh(W_q h + b_q) \quad (6)$$

where $W_q$, and $b_q$ are the weight matrix and bias vector to be trained.
Finally, a GRU is used to decode $h_q$ and $h_{q_1}$ to generate two sub-questions $q_1$ and $q_2$ word by word as the final output of the model:

$$q_i = decoder(h_{q_i})$$  

(7)

In the above process, an attention mechanism (Fig 5c) [21] is incorporated to allow the decoder referencing to the relevant words in the input question when generating each word in the decomposed sub-question. It further enhances the semantic relevance between the input and output questions. At the time step $t$, the attention layer first calculates the attention weights by considering current hidden state $h_t$ in the decoder and all the hidden states of the encoder $h_{encoder} = [h_1, h_2, ..., h_n]$:

$$a_t = softmax(h_t^\top h_{encoder})$$  

(8)

The contextual vector for the input question $v_{ctx}$ is then computed as the weighted average over all the encoder states. After that, the attention layer concatenates $v_{ctx}$ and $h_t$ to produce a new hidden state $h'_t$ for further predicting next word in the decoder:

$$h'_t = tanh(W_{att}v_{ctx}; h_t)$$  

(9)

where $W_{att}$ is the weight matrix to be trained.

To design a robust model, we have to consider the situation when decomposing a question about a dataset outside the scope of the training corpus. In this case, an attention mechanism is not enough as it is difficult to predict an unseen word, e.g., the column name in the new dataset, when generating a sub-question. To address this issue, we integrated the copying mechanism (Fig 5d) [27] in the model, which generalized the model by selectively copying some unseen words directly from the input (either question or data columns) when generating a sub-question. Intuitively, it estimates the probability $p_c \in [0,1]$ of using a word copied from the question/data column instead of using a word generated by decoder when produces a new sub-question. Formally, $p_c$ is computed by combining the current hidden state $h_t$ in the RNN model, the contextual vector $v_{ctx}$ from the attention mechanism, and the last generated word $w_{i-1}$ together:

$$p_c = sigmoid(v^1_c h_t + v^2_c v_{ctx} + v^3_c w_{i-1})$$  

(10)

where $v_c$ is the trainable weighting vector that transform the above three vectors into a single value to predict the probability.

To encourage the output of the decomposition model as identical as possible with the target sentences in our training corpus, the model is trained by minimizing the word-level negative log likelihood loss [19]:

$$Loss = - \sum_{i=1}^{n} \log p(t_i | t_1, ..., t_{i-1}, Q_z)$$  

(11)

where $t_i$ is the current reference word in the target sentence. Given the previous words $t_1, ..., t_{i-1}$ and the input question $Q_z$, this loss function tends to maximize the probability of the reference word $t_i$ as the prediction for current word.

Implementation: The decomposition model was implemented in PyTorch [41]. Both encoder and decoder takes a two-layer GRU with 0.1 dropout rate for avoid over-fitting. The word embedding size and hidden size are both set to 256. The maximum length of the input sentence is 60 words. All the training parameters are initialized and updated via the Adam optimizer [29], with a learning rate of 0.0001. The model was trained on a Nvidia Tesla-V100 (16GB) graphic card.

4.4 Training Corpus

To train our model, we prepared a new table-based question decomposition corpus [3] with the help of 700 English native speakers from the crowdsourcing platform [3]. The corpus consists of 9,071 high-level questions including 3,492 Type-I questions and 5,579 Type-II questions. Two low-level questions were prepared as the decomposition results for each of these high-level questions, i.e., 18,142 low-level questions were prepared. In our corpus, we guarantee each low-level question corresponds to a low-level analysis task to ensure the question can be answered by at least one data fact. In general, we prepared the corpus via four steps: (1) selecting a set of meaningful data tables in various domains based on which high-level questions will be prepared; (2) generating high-level questions with all type of structures by a computer program; (3) manually polishing the machine-generated questions to reach the standard of natural language via the crowdsourcing platform; (4) eliminating the low-quality questions.

Table Selection: We collected 70 tabular datasets in different domains from Kaggle and Google dataset search. These datasets were further filtered based on three criteria: (1) containing meaningful data column headers; (2) having sufficient data columns and diverse column types to support all types of questions; and (3) containing informative data insights. As a result, 26 data tables were selected. Each of them has a meaningful column header and contains at least one numerical, one temporal, and one categorical field. All of them were tested by an online auto-insights tool to make sure meaningful data insights could be discovered from the data. We also make the size of the data diverse, the number of rows ranges from 26 to 86,454 (mean 7,067); the number of columns ranges from 4 to 16 (mean 9).

A Question Generation to generate a high-level question, we created a set of random facts and select the insightful ones based on the methods introduced in [50]. After that, we enumerated the facts to generate meaningful fact combinations that are potential answers to a high-level question based on the methods introduced in [34][48]. Finally, the fact combinations are translated into a high-level question based on over 200 manually prepared question templates. We traversed all 26 selected data tables and generated 5,500 Type-I questions and 7,500 Type-II questions, respectively.

Specifically, to create the Type-I questions, we selected data facts via three question reasoning methods, i.e., comparison, intersection, and bridging, as introduced in [34]. In particular, the “comparison” type of questions compare facts within different data scopes based on the same measurement. For example, the question “What are the differences between USA and China in terms of car sales trend in recent 5 years?” is a comparison between the data respectively collected in USA and China under the same measurement of sales value. The “intersection” type of questions seek for data elements that satisfies the conditions specified by different data facts. For example, the question “which areas have a increasing temperature and a decreasing population” seeks for the areas (data elements) that satisfies the conditions specified by two trend facts, i.e., increasing temperature and decreasing population. Finally, the “bridging” type of questions asks for a data fact that satisfies a prior condition specified by another fact. For example, the question “what is the trend of temperature in an area with a decreasing population” asks for a trend fact (trend of temperature) based on a condition specified by another trend fact (decreasing population).

To generate Type-II questions, we consider three forms of high-level fuzzy questions: (1) the questions without mentioning an analysis task (i.e., no fact type) such as “How about BMW?”; (2) the questions without mentioning the aspect to be estimated (i.e., no measure) such as “How about the recent trend of BMW?”; (3) the questions without mentioning data divisions (i.e., breakdown methods) such as “What is the distribution of BMW’s sales record?” Obviously these questions do not have a unique answer. For example, the third question contains two situations, i.e., distributions over time and over regions. Therefore, we choose all the facts that potentially answers such a high-level question to help generate questions in these three forms.

Question Rephrasing: To produce high-quality nature language questions, we employed a group of native English speakers to manually rephrase and polish the generated questions through a crowdsourcing platform [2]. To this end, an online system was developed. It splits the job by randomly allocate 50 machine-generated questions to each participant who was asked to fix the grammar errors and polish the questions into a nature language representation without changing their

2The dataset will be released after the review.

https://www.prolific.co/

https://lite.datacalliope.com/
After that, from each of the 50 questions processed by a participant, the first thing to retrieve an answer fact to find out the answers. To balance performance and efficiency, we focus to extract data facts in Talk2Data, the answer extraction module searches the data space \(X\) to indicate the type of fact \(f_i\) given a low-level question \(q_i\). We fine-tuned the BERT model by combining an additional classification layer and trained it on our corpus to classify the input low-level question:

\[
t = \text{softmax}(W u), \quad u = BERT(q_i)
\]

where \(u\) is the intermediate vector encoder generated by the BERT model and \(W\) is the trainable matrix in the classification layer. The output \(t\) is a ten-dimensional one-hot vector that represents the type of answer fact \(f_i\) to the input low-level question \(q_i\).

### 5.2 Fact Search

Given a preferred fact type \((t_i)\), we employ the beam search algorithm \([38]\) to determine the rest of fact fields, i.e., subspace \((s_i)\), breakdown \((b_i)\), measure \((m_i)\), and focus \((x_i)\) by searching through the entire data space \(X\). In particular, we use semantic similarities between the resulting facts \(f_i\) and the input question \(q_i\) as the heuristic function to guide the searching process with the goal of retrieving facts that are most relevant to the question. As shown in Fig [5] and summarized in Algorithm [1] the algorithm explores the data space \(X\) to examine a number of candidate choices for each fact field step by step via a search tree \(T\). In each searching step, the algorithm chooses a candidate value (e.g., a data column) for the corresponding fact field that makes the fact having the highest reward. Finally, the fact is determined by a path from the root to a leaf in the search tree.

The search tree \(T\) is gradually generated through a searching process as described in Algorithm [4]. In particular, the algorithm takes a low-level question \(q_i\), a fact type \(t_i\), a tabular data \(X\), and a beam width \(k\) (the number of selected facts at each round) as the input and automatically generates a set of data facts \(\mathcal{F}\) that are most relevant to the question. At the beginning, the type \(t_i\) initializes a data fact \(f_{\text{initial}}\) to confirm the rest fields in the fact, and use \(f_{\text{initial}}\) as the root of the \(T\) (line 1, Fig [5]b). In next, the algorithm generates data facts by iteratively searching the rest fields in facts via three major steps: selection, expansion, ranking. The first step selects \(T_{\text{top}}\) that contains the highest \(k\) facts ranked in \(T\), from which the next expansion step will be performed (line 3, Fig [5]b1). The second step expands the \(T\) by creating a set of data facts. (line 4 - 8, Fig [5]b2). In this step, the new data facts are generated by filling

| Question Type | Method | % | High-level Question | Decomposed Question |
|---------------|--------|---|---------------------|---------------------|
| Type-I        | Comparison | 22.8 | Which genre has more user reviews, fiction book or non-fiction book? | (1) How many reviews the fiction book has? (2) How many reviews the non-fiction book has? |
|               | Intersection | 10.3 | Which book is expensive and well-regarded? | (1) Which book has a review higher than average? (2) Which book has a price higher than average? |
|               | Bridging | 28.4 | In the year with most reviews, what is the distribution of price over different genre? | (1) Which year has the highest/lowest reviews? (2) What is the overall distribution of price over genre? |
| Type-II       | No fact type | 20.6 | Show me some information about book price in the different genre. | (1) What are the differences in price between each genre? (2) Which genre has the highest price? |
|               | No measure | 5.82 | Which genre of book is an outlier compare with other books? | (1) Which genre of book has an anomaly user rating? (2) Which genre of book has an anomaly reviews? |
|               | No breakdown | 8.1 | What is the outlier of user rating? | (1) What is the outlier of user rating over different years? (2) What is the outlier of user rating over different genre? |

### Table 1. Examples of high-level questions and the corresponding decomposed low-level questions. % shows the proportions of the questions generated by the different methods.
Algorithm 1: Answer Fact Searching

Input: \( q_i, t, X, k \)
Output: \( \mathcal{F} = \{ f_1, f_2, \ldots, f_k \} \)

1. \( f_{\text{initial}} \leftarrow \{ t, *, *, *, * \}; T \leftarrow \{ f_{\text{initial}} \}; \mathcal{F} \leftarrow [] \); // Gradually determine the rest of the fact fields, following the order of measure, breakdown, subspace, focus

2. for \( p_i \in \{ \text{measure}, \text{breakdown}, \text{subspace}, \text{focus} \} \) do
   3. \( T_{\text{top}} \leftarrow \text{select}(T[k]) \); /* 1.selection */
   4. \( f_j \in T_{\text{top}} \) do
      5. if \( p_i \) is not required in \( f_j \) then continue;
   6. // Expand the \( T \) by creating a set of data facts, each fact is generated by filling the field \( p_i \) in \( f_j \) with a candidate value from \( X \)
   7. \( T \leftarrow \text{expand}(f_j[p_i, X]) \);
   8. end /* 2.expansion */
   9. \( T \leftarrow \text{rank}(T[q_i]) \); /* 3.ranking */

10. end // After all the fact fields are filled, the top \( k \) facts ranked in \( T \) are identified as the most relevant facts for the question \( q_i \)
11. \( \mathcal{F} \leftarrow \text{select}(T[k]) \);
12. return \( \mathcal{F} \);

the field \( p_i \) in \( f_j \) with candidate values from \( X \). The third step ranks the new facts in \( T \) based on the semantic similarities between \( f_j \) and question \( q_i \) (line 10). After facts are complete, the top \( k \) facts ranked in \( T \) are identified as the most relevant facts for the question \( q_i \) (line 12).

5.3 Fact Ranking

To retrieve a set of facts \( \mathcal{F} \) that are most relevant to the input low-level question \( q_i \), in each round of expansion, the facts \( f_j \) in \( T \) are ranked by their semantic similarity between \( f_j \) and \( q_i \). As facts \( f_j \) are in the form of 5-tuple, we first use hand-written templates to transform facts into machine-generated questions. If the facts \( f_i \) are not complete, questions will be generated based on existing fields. Then we employ the Sentence-BERT [43] to project machine-generated questions and the input question into the same vector space and use the cosine-similarity between corresponding vectors to estimate their semantic similarity.

6 User Interface and Visualization

In this section, we introduce the representation module of the Talk2Data system. We demonstrate the design of the system’s user interface and the corresponding interactions. After that, we introduce how the data facts are visually represented by a library of annotated charts and arranged in order to answer the input question.

6.1 User Interface and Interactions

The user interface of the Talk2Data system consists of two views: (1) the data view (Fig. 6(a)) and (2) the answer view (Fig. 6(b)). In particular, the data view is designed to illustrate the raw data to users so that they could initiate a question. In particular, the data is shown in a data table (Fig. 6(a1)) whose columns are colored by the corresponding data types. A question panel (Fig. 6(a2)) is also provided in the view to display potential questions that could be asked about the data. When a data column is selected, the question list will be updated accordingly to show questions that are only relevant to the selected column. The answer view (Fig. 6(b)) represents data facts that answer the question via a library of annotated charts, the charts are arranged in order to facilitate the interpretation and narration of the answers. In particular, in this view, user’s question is represented as the title of view (Fig. 6(b1)), and decomposed questions are shown as sub-titles in each section that are answered by data facts (Fig. 6(b2)). Each data fact is visualized by an annotated chart with a narrative caption.

A floating tool bar (Fig. 6(a3, b3)) is designed and placed at the bottom of both views, through which users can upload the data, ask a question via both voice and text input, edit the input mode for editing the answers, and switch between the data and answer views. Users can also enter the full-screen mode or covert the results into a PDF report by clicking the buttons in the upper-right corner (Fig. 6(b4)).

6.2 Annotated Chart Library for Tabular Data

In data storytelling, annotations in charts will help emphasize the information and avoid ambiguity [30]. Therefore, we design a library of annotated statistical charts to display the data facts that answer the question with the goal of helping with the answer narration and interpretation. Our chart library consists of 5 types of basic statistic diagrams (bar chart, line chart, pie chart, area chart, and scatter plots) that are frequently used in data stories as summarized in [50]. To design the annotations, we further investigated a large number of relevant designs by exploring the design of the text input, frequently used in over 200 data videos and over 1500 info-graphics. As a result, annotations involving text, colors, shapes, pointers, lines are designed for showing values, illustrating the trends and relations, highlighting anomalies and extremes, emphasizing differences and ranks, and differentiate categories and proportions. Applying annotations in the aforementioned 5 types of charts to represent different narrative semantics gives us 15 different annotated charts(Fig. 7). For example, we use dash-lines in a bar chart to emphasize difference but use trending lines in bar chart to illustrate trend, which result in two different annotated charts.

Despite the above annotations, caption is another crucial component in each of the annotated charts. It usually describes the important data patterns in a nature language to help users quickly capture the information shown in the chart. To generate the caption, we adopt the sentence template for each type of fact introduced in [50]. Considering these templates may generate problematic descriptions that have grammar errors, our system enables a free editing function, through which users can easily edit the captions to fix the errors when necessary.

6.3 Answer Facts Layout

We display the answers to the input question in the form of a dashboard that could be easily displayed on a big screen to facilitate online discussions about the data in real-time. The annotated charts are arranged in order to facilitate reading and answer narration. In particular, we divide the screen into several regions and allocate to decomposed questions. Within each region, we arrange the charts in order according to their relevance to the corresponding questions, and place them one by one from left to right and top to bottom to facilitate reading. The size of each chart is determined by their relevance score to the question.
We estimate the design of the system via an example case, quantitative evaluations and a controlled user study.

### 7.1 Case Study
We performed a case study by inviting an expert from a business school to explore and analyzing a marking dataset about car sales records. The dataset consists of four dimensions: year, sales value, model, and brand (Fig. 4(a)). The user started with a high-level fuzzy question (i.e., Type-II) “How is the sales trend when the brand is Volkswagen.” The system automatically decomposed the question into three relevant low-level questions: “which category has the highest sales?” (Q1.1), “what is the sales over years?” (Q1.2), and “what is the overall value of the sales?” (Q1.3), which are answered by three annotated charts showing the best selling model, the overall sales trend, and the total sales value as shown in Fig. 4(c-1).

Having the above overview of the data, the user would like to dig deeper into the data. He asked “does any brand sell a lot and have an increasing trend?” (i.e., Type-I). The system resolves the question into three relevant low-level questions: (1) “what is the trend of sales?” (Q2.1), (2) “which brand has the highest/lowest sales?” (Q2.2), and (3) “which brand has an increasing trend of sales?” (Q2.3). The answers to these questions are shown in a group of charts as illustrated in Fig. 4(c-2), which respectively illustrates the sales trend, showing the highest sales record among different brands and the sales trend of each brand.

### 7.2 Quantitative Evaluation

#### Decomposition Quality
We estimate the quality of the question decomposition results based on the testing corpus via two frequently used metrics in nature language processing, i.e., BLEU [59] and METEOR [8]. These metrics are originally designed to estimate the quality of sentence translation. In particular, BLEU estimates the word-level translation precision based on the number of exactly matched words between the translated sentences and the ground-truth. METEOR computes a weighted F1-score to estimate the translation quality by comparing the translated sentences and ground-truth based on WordNet [33]. These metrics are verified to be able to provide estimations that are consistent with humans’ judgments [79]. In our experiment, we use these metrics to estimate the decomposition quality by comparing the decomposed sub-questions to the corresponding targets in the testing corpus.

Due to the lack of similar techniques, we estimate the performance of the proposed decomposition model by comparing it to three simplified versions that respectively have (1) no copying mechanism, (2) no copying and attention mechanisms, and (3) no copying, attention, and question type classification components. All these models were trained based on the question decomposition corpus under the same parameters settings. In particular, the training set, validation set, and evaluation set respectively takes 80%, 10%, and 10% of the corpus. The evaluation results are summarized in Table 2 which show that our designs of the key components, i.e. question type classifier, attention and copying mechanism indeed improve the performance of the question decomposition model. The BLEU and METEOR values also are equivalent to that of a high-quality sentence rewriting [60].

#### Accuracy
To estimate the accuracy of the answers given by the Talk2Data system, we computed the precision and recall and draw the corresponding ROC curve based on the following definition:

\[
\text{precision} = \frac{|\text{relevant facts} \cap |\text{retrieved facts}|}{|\text{retrieved facts}|}
\]
\[
\text{recall} = \frac{|\text{relevant facts} \cap |\text{retrieved facts}|}{|\text{relevant facts}|}
\]  
(15)

#### Table 2. Performance Evaluation of the Decomposition Model

| Models                      | BLEU  | METEOR |
|-----------------------------|-------|--------|
| Decomposer                  | 23.88 | 24.02  |
| Decomposer + Classifier     | 25.23 | 25.73  |
| Decomposer + Classifier + Attention | 25.56 | 26.09  |
| Decomposer + Classifier + Attention + Copying | 26.22 | 27.55  |
where the \{retrieved facts\} are the top-k data facts that retrieved from the searching results to answer a question. Here, k is determined by the maximum number of charts that could be displayed on the screen. In our implementation, k = 12. The \{relevant facts\} indicate a collection of data facts that are relevant to the ground truth answers in a search. The relevance is computed based on the cosine-similarity between two sentence vectors generated by the Sentence-BERT encoder \[44\]. A threshold \(R_t\) is given to determine which set of facts are relevant to the answers, which also gives different value points in the ROC curve.

Our experiment was performed based on three real-world datasets, denoted as D1, D2, D3. 200 high-level questions were asked based on each of the dataset. For each question, the system decomposed the question and searched from the answer within 2 minutes. The averaged precision and recall were computed based the searching results and reported in the ROC-curves as shown in Fig.\[8\]. In particular, our system generally performs better on smaller datasets with lower dimensions (D1, \(F_1 = 0.71, R_t = 0.55\) when compared to datasets that have more records (D2, \(F_1 = 0.657, R_t = 0.6\)) and have higher dimensions (D3, \(F_1 = 0.638, R_t = 0.65\)).

### 7.3 User Study

To estimate the usability of the system, we conducted a controlled within-subject study with 20 participants to make a comparison between Talk2Data and a baseline system developed based on NL4DV and vega-lite charts. The participants (13 female, 7 male, between 21 and 28 years old (M = 24.8, SD = 1.94)) are university students major in design and literature. They have limited knowledge about data analysis.

Two real datasets were used for the study. The first one describes 549 Amazon bestselling books (rows) from six dimensions (columns) including book title, rating, number of reviews, published year, price, and genre. The second dataset describes 275 car sales records from four dimensions including the sales value, brand, model, and the year. These two datasets were used in both the Talk2Data and the baseline system during the study in a counterbalanced order. During the study, we first introduced the systems and let the participants to try it by their own. After the users were getting familiar with systems, they were asked to finish six tasks by asking relevant questions by their own. Three of these tasks were low-level ones such as “find the distribution of ratings over books” but the other three were high-level ones such as “find the most popular author”. During the experiment, we recorded the number of questions asked by a user to finish each task in each system. This number were together with the accuracy to estimate their performance. Finally, the participants were also asked to finish a post-study questionnaire. The study results are reported as follows:

**Accuracy.** As shown in Fig.\[9\](a), Talk2Data (M=95\%, SD=0.16) and the baseline (M=95\%, SD=0.17) had a similar accuracy when finding answers to low level questions. However, our system (M=87.6\%, SD=0.17) significantly outperformed the baseline system (M=67.5\%, SD=0.28) in case of resolving high-level questions based on the paired-t test (t(19) = 4.08, p < 0.01).

**Efficiency.** We compared the averaged number of questions that users need to finish each of the tasks to demonstrate a system’s efficiency. As shown in Fig.\[9\](b), when solving low-level tasks, users explored a similar number of questions when using Talk2Data (M = 1.17, SD = 0.49) and the baseline (M = 1.32, SD = 0.76) system. However, for high-level tasks, users obviously tend to ask more questions when using the baseline system (M = 2.63, SD = 1.53) comparing to that of Talk2Data (M = 1.62, SD = 1.11). The difference is significant regarding to the paired-t test (t(59) = 4.4, p < 0.01).

**Feedback.** The results (Fig.\[9\](c)) of our post-study questionnaire showed that all the users preferred our system. Most of them mentioned “Talk2Data is a useful tool”, “it can greatly save one’s efforts when exploring the data”, and “the system is easy to use”. Many users also mentioned “dividing a complex question into simple ones and answer them one by one is an intuitive and effective way to solve the problem”.

Fig. 8. The precision-recall (ROC) curves of the variations of the Talk2Data system on three datasets with different sizes.

Fig. 9. The results of the user study: (a) the accuracy, (b) the averaged number of questions to finish each of the tasks, and (c) the ratings from different criteria based on a 5-point Likert scale, where 5 is the best and 1 is the worst.
9 Conclusion

We present Talk2Data, a data-oriented online question and answering system that supports answering both low-level and high-level questions. The system employs a novel deep-learning based question decomposition model to resolve a high-level question into a series of relevant low-level questions, and a search algorithm to extract the data facts that are most relevant to each of low level questions. To visualize the data facts, we designed a set of annotated and captioned visualization charts to support interpretation and narration. The proposed technique was evaluated via case studies, performance validation, and a controlled user study. The evaluation showed the power of the Talk2Data system and revealed several limitations of the current system, which will be addressed in the future.

References

[1] Microsoft power bi q&a. https://powerbi.microsoft.com [Online; accessed 11-March-2021].

[2] Prolific. https://prolific.co/ [Online; accessed 11-March-2021].

[3] Tableau ask data. https://www.tableau.com/products/new-features/ask-data [Online; accessed 11-March-2021].

[4] R. Amar, J. Eagan, and J. Stasko. Low-level components of analytic activity in information visualization. In IEEE Symposium on Information Visualization, pp. 111–117. IEEE, 2005.

[5] A. Amini, S. Gabriel, S. Lin, R. Koncel-Kedziorski, Y. Choi, and H. Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, pp. 2357–2367. Association for Computational Linguistics, Minneapolis, Minnesota, 2019.

[6] J. Aurisano, A. Kumar, A. Gonzalez, J. Leigh, B. DiFugenio, and A. Johnson. Articulate2: Toward a conversational interface for visual data exploration. In IEEE Visualization, 2016.

[7] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In International Conference on Learning Representations, 2015.

[8] S. Banerjee and A. Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, pp. 65–72. Association for Computational Linguistics, Ann Arbor, Michigan, 2005.

[9] J. Bao, N. Duan, Z. Yan, M. Zhou, and T. Zhao. Constraint-based question answering with knowledge graph. In Proceedings of International Conference on Computational Linguistics, pp. 2503–2514. The COLING 2016 Organizing Committee, Osaka, Japan, 2016.

[10] J. Bao, N. Duan, M. Zhou, and T. Zhao. Knowledge-based question answering as machine translation. In Proceedings of the Association for Computational Linguistics, pp. 967–976. Association for Computational Linguistics, Baltimore, Maryland, 2014.

[11] J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on freebase from question-answer pairs. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1533–1544. Association for Computational Linguistics, Seattle, Washington, USA, 2013.

[12] A. Bordes, N. Usunier, S. Chopra, and J. Weston. Large-scale simple question answering with memory networks. arXiv preprint arXiv:1506.02075, 2015.

[13] W. Chen, H. Zha, Z. Chen, W. Xiong, H. Wang, and W. Y. Yang. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1026–1036. Association for Computational Linguistics, Online, 2020.

[14] M. Cho, R. K. Amplayo, S.-w. Hwang, and J. Park. Adversarial tableqa: Attention supervision for question answering on tables. In Asian Conference on Machine Learning, pp. 391–406. PMLR, 2018.

[15] K. Cox, R. E. Grimm, S. L. Hilhino, L. J. Jagadeesan, and D. Mantilla. A multi-modal natural language interface to an information visualization environment. International Journal of Speech Technology, 4(3):297–314, 2001.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the Association for Computational Linguistics, pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota, 2019.

[17] M. Ding, C. Zhou, Q. Chen, H. Yang, and J. Tang. Cognitive graph for multi-hop reading comprehension at scale. In Proceedings of the Association for Computational Linguistics, pp. 2694–2703. Association for Computational Linguistics, Florence, Italy, 2019.

[18] S. Eduvov, M. Ott, M. Auıl, D. Grangier, and M. Ranzato. Classical structured prediction losses for sequence to sequence learning. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics, pp. 355–364. Association for Computational Linguistics, New Orleans, Louisiana, 2018.

[19] Y. Feldman and R. El-Yaniv. Multi-hop paragraph retrieval for open-domain question answering. In Proceedings of the Association for Computational Linguistics, pp. 2296–2309. Association for Computational Linguistics, Florence, Italy, 2019.

[20] T. Gao, M. Dontcheva, E. Adar, Z. Liu, and K. G. Karahalios. Datatone: Managing ambiguity in natural language interfaces for data visualization. In Proceedings of the Annual ACM Symposium on User Interface Software & Technology, p. 489–500. Association for Computing Machinery, New York, NY, USA, 2015.

[21] J. Gu, Z. Lu, H. Li, and V. O. Li. Incorporating copying mechanism in sequence-to-sequence learning. In Proceedings of the Association for Computational Linguistics, pp. 1631–1640. Association for Computational Linguistics, Berlin, Germany, 2016.

[22] H. Hashemi, M. Aliannejadi, H. Zamani, and W. B. Croft. Antique: A non-factoid question answering benchmark. In J. M. Jose, E. Yilmaz, M. Magalhães, P. Castells, N. Ferro, M. J. Silva, and F. Martins, eds., Advances in Information Retrieval, pp. 166–173. Springer International Publishing, Cham, 2020.

[23] D. Hoogeveen, K. M. Verspoor, and T. Baldwin. Cqadupstack: A benchmark data set for community question-answering research. In Proceedings of the Australasian Document Computing Symposium. Association for Computing Machinery, New York, NY, USA, 2015.

[24] E. Hoque, V. Sethur, M. Tory, and I. Dykeman. Applying pragmatics principles for interaction with visual analytics. IEEE Transactions on Visualization and Computer Graphics, 24(1):309–318, 2017.

[25] M. Iyyer, J. Boyd-Graber, L. Claudino, R. Socher, and H. Daume III. A neural network for factoid question answering over paragraphs. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 633–644. Association for Computational Linguistics, Doha, Qatar, 2014.

[26] S. K. Jauhar, P. Turney, and E. Hovy. Tables as semi-structured knowledge for question answering. In Proceedings of the Association for Computational Linguistics, pp. 474–483. Association for Computational Linguistics, Berlin, Germany, 2016.

[27] Z. Jia, A. Abujabal, R. Saha Roy, J. Strötgen, and G. Weikum. Temptuations: A benchmark for temporal question answering. In Companion Proceedings of the The Web Conference, p. 1057–1062. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2018.

[28] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Proceedings of the Association for Computational Linguistics, pp. 1601–1611. Association for Computational Linguistics, Vancouver, Canada, 2017.

[29] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[30] B. Lee, N. H. Riche, P. Isenberg, and S. Carpendale. More than telling a story: Transforming data into visually shared stories. IEEE Computer Graphics and Applications, 35(5):84–90, 2015.

[31] T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1412–1421. Association for Computational Linguistics, Lisbon, Portugal, 2015.

[32] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In Proceedings of the Association for Computational Linguistics, pp. 55–60. Association for Computational Linguistics, Baltimore, Maryland, 2014.

[33] G. A. Miller. Wordnet: a lexical database for english. Commun. ACM, 38(11):39–41, 1995.

[34] S. Min, V. Zhong, L. Zettlemoyer, and H. Hajishirzi. Multi-hop reading comprehension through question decomposition and rescoring. pp. 6097–6108. Association for Computational Linguistics, Florence, Italy, 2019.

[35] A. Narechania, A. Srinivasan, and J. Stasko. NL4dv: A toolkit for generating analytic specifications for data visualization from natural language.
