NL4DV: Toolkit for Natural Language Driven Data Visualization

Arjun Srinivasan* John Stasko†

Georgia Institute of Technology

ABSTRACT

Developing natural language interfaces for visualization systems is a challenging task and requires system developers to spend time and effort on implementing Natural Language Processing (NLP) components necessary to convert natural language queries into visualizations. Especially for developers without a background in NLP, this learning curve can be even more challenging and time consuming. We are developing the Natural Language Driven Data Visualization (*NL4DV*) toolkit that provides high-level functions developers can use to create natural language-driven data visualization systems.

1 INTRODUCTION

The idea of using natural language as a querying interface for visualization systems is becoming increasingly popular. Articulate [8] is an example of a system presenting a natural language interface for visualization. Articulate maps user queries to tasks and uses these tasks in combination with data attributes to generate required visualizations. More recently, Gao et al. developed DataTone [3], a mixed-initiative system targeted at helping users resolve ambiguity in natural language queries. DataTone presents "ambiguity widgets" that assist users to iteratively construct visualizations. Commercial applications like IBM Watson Analytics[1] and Microsoft Power BI[2] also present natural language interfaces for visualization.

Developing visualization tools driven by natural language is a challenging task, however, and it requires implementing not only the user interface and visualization components but also the NLP components necessary to convert a query to a visualization. Existing NLP toolkits (e.g., [5, 4]) provide functionality such as Parts-of-Speech (POS) tagging and entity extraction that can be used to implement aspects of these components. While such toolkits are useful for general-purpose NLP tasks and provide a wide repertoire of functionality, they are fairly complex and require developers to understand NLP concepts for effective use. Other existing research in the space of natural language querying of databases (e.g., [7, 2]) explores how keywords can be used to extract relevant information from a query in the context of a database. While these systems work well for their targeted users, they expect fairly structured queries and do not support unstructured exploratory questions.

Even with this body of existing work, converting a natural language query to a set of visualizations remains a challenging task. It typically involves processing a query to identify a set of data attributes and analytical tasks mentioned in the query. These attributes and tasks then need to be mapped to relevant visualizations that can be used to answer the input question. Implementing these steps is generally time consuming and involves a steep learning curve, especially for developers without a NLP background.

*e-mail: arjun010@gatech.edu

†e-mail: stasko@cc.gatech.edu

[1] http://www.ibm.com/analytics/watson-analytics/

[2] https://powerbi.microsoft.com/en-us/

We are creating a toolkit called NL4DV to provide developers and designers of visualization systems with high-level functions that they can use to add natural language query interfaces as extensions to their existing systems or build systems entirely driven by natural language.

2 NL4DV

Figure 1 highlights (in gray) the four high-level components and some of their intermediate inputs and outputs (in white) that are required to implement a natural language driven visualization system. NL4DV is implemented in Python and is provided as a python package. As shown in Figure 1, it comes with a built-in query processor and visualization recommendation engine which are the core components of this implementation pipeline.

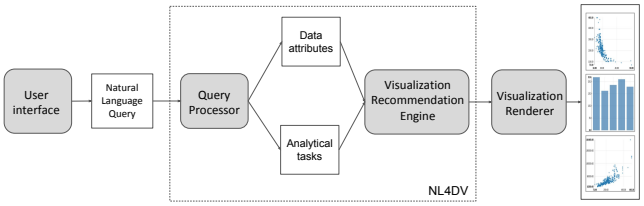

Figure 1: Typical components of a natural language driven visualization system

To operate, NL4DV simply requires a data source (currently supported formats include CSV, TSV, and JSON files) and a natural language query (as a string). The toolkit then returns a list of data attributes (*dataAttributesMap*) and analytical tasks (*taskList*) being referred to in the query, and a ranked list of visualizations (*recommendedVisualizations*) that can be used to answer the question most effectively. These responses can then be used by the developer to render the visualizations or update the user interface depending on the application's context.

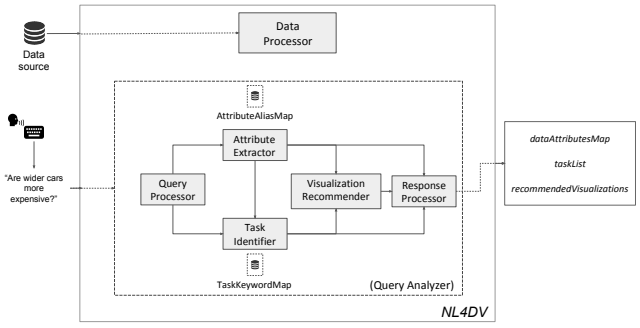

Figure 2: NL4DV architecture

2.1 Components

Figure 2 highlights NL4DV's two primary components, the data processor and the query analyzer. The query analyzer further en-

capsulates five modules. A brief description of how these components and modules work together to process a query follows:

Data Processor. On initializing NL4DV with a data source, this component parses the input data source to extract details about individual attributes (e.g. data type, domain etc.). This information is used later by the task identifier and visualization recommendation modules.

Query Processor. The goals of this module are to process the input query to generate a list of *n*-grams that constitute the sentence and perform Parts-Of-Speech (POS) tagging on the input query[3]. *n*-grams are generated with *n* ranging from 1 (single word) to *k* (number of words in the query).

Attribute Extractor. The attribute extractor uses the *n*-grams from the query processor to identify data attributes mentioned in the input query. To match *n*-grams to attributes, the module uses a combination of string similarity score (using Levenshtein distance) and the distance between the stemmed versions of the strings in the wordnet graph [6]. When comparing *n*-grams to attributes, this module also leverages the optional *AttributeAliasMap* provided by the developer. This map essentially holds aliases for attribute names (e.g. "MPG" for "Miles Per Gallon"). To support follow-up questions, this module persists the attributes identified in a query until it encounters another query containing new attributes.

Task Identifier. This module generates a list of analytical tasks implicitly or explicitly stated in the input query. NL4DV currently supports identification of the ten tasks (*Retrieve Value*, *Filter*, *Compute Derived Value*, *Find Extremum*, *Sort*, *Determine Range*, *Characterize Distribution*, *Find Anomalies*, *Cluster*, and *Correlate*) in the taxonomy proposed by Amar et al [1]. To identify tasks, the module uses a combination of pre-defined keywords, POS-tags, and a dependency parser to map tasks to attributes. Each pre-defined keyword maps to one or more tasks. These keywords are generated using questions collected by Amar et al [1] and stored in the *TaskKeywordMap*. This map can be extended by the developer based on the context of a dataset to map specific keywords to tasks. Since tasks can be implicity stated, there is uncertainty associated with identifying them. This uncertainty is represented using a confidence score (ranging from [0,1]) associated with each task.

Visualization Recommender. This module uses the attributes and tasks identified to generate and rank possible visualizations that are relevant to the input question. Currently supported visualizations include bar chart, line chart, pie chart, scatterplot, and histogram. The module first lists all combinations of attributes and possible transforms. These are then scored (in the range [0,1]) based on how relevant and useful each of them are given the tasks at hand (e.g., for a correlation task involving two numeric attributes, a scatterplot will be assigned the highest score). Each visualization is represented as a simple JSON object that can be parsed by the developer to render visualizations on the front end.

Response Processor. This module takes the output from the attribute extractor, task identifier, and the visuailzation recommender and combines them into a single object. This module also adds a boolean *isFollowUpQuestion* flag to the output depending on if a question is identified as a follow-up question or not. It then returns the generated object as the output to the calling function.

Figure 3 summarizes the output from the query processor, attribute extractor, task identifier, and the visualization recommender modules for the sample query "Are wider cars more expensive?" in the context of a cars dataset.

3 Status and Future Work

We have provided a trial version of the NL4DV toolkit to graduate students to use as a part of their projects. Some students have already built natural language driven visualization systems from scratch using it. Other students are experimenting with using the toolkit to add an optional natural language query interface to existing applications.

[3]Supported POS-tags are listed at `http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`

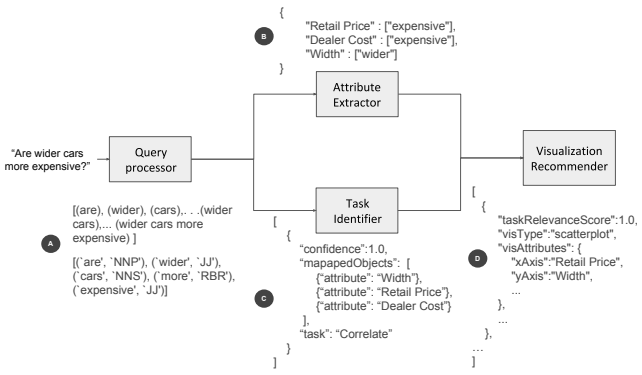

Figure 3: Query Analyzer sample outputs. (A) *n*-grams and POS-tagged list[3] (B) *DataAttributesMap*, (C) *TaskList*, (D) *RecommendedVisualizations*

We are encouraged by the positive feedback this initial version of the toolkit has received. We are currently working on setting up experiments to evaluate the accuracy of the toolkit in identifying attributes and tasks across multiple datasets and domains. Based on some of the initial feedback, we also are exploring ways to improve the query analyzer by using more advanced dependency parsing techniques. Further, we are delving into the idea of adding a conversational assistant to the toolkit. This would allow the toolkit to summarize the results and also provide human-readable responses in cases where it is unable to parse a query. We are exploring the use of Artificial Intelligence Markup Language (AIML) as a potential way of implementing this "brain" for the assistant.

To support future research and experimentation in the space of natural language interfaces for visualization, we will soon be making NL4DV freely available as open-source software. We hope this work will provide valuable building blocks for natural language driven visualization research and development.

References

[1] R. Amar, J. Eagan, and J. Stasko. Low-level components of analytic activity in information visualization. In *Proceedings of IEEE InfoVis '05*, pages 111–117, 2005.

[2] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. Natural language interfaces to databases–an introduction. *Natural language engineering*, 1(01):29–81, 1995.

[3] 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 ACM UIST '15*, pages 489–500, 2015.

[4] E. Loper and S. Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1*, pages 63–70, 2002.

[5] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[6] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[7] G. Orsi, L. Tanca, and E. Zimeo. Keyword-based, context-aware selection of natural language query patterns. In *Proceedings of the 14th International Conference on Extending Database Technology*, pages 189–200. ACM, 2011.

[8] Y. Sun, J. Leigh, A. Johnson, and S. Lee. Articulate: A semi-automated model for translating natural language queries into meaningful visualizations. In *Smart Graphics*, pages 184–195, 2010.