ISLAMIC UNIVERSITY OF TECHNOLOGY

Development Of An Explainable Natural Language Query Driven Data Visualization System

By

MD. Hamjajul Ashmafee (171041012)

A thesis submitted in partial fulfilment of the requirements for the degree of M.Sc. in Computer Science and Engineering

Academic Year: 2017-2018

Department of Computer Science and Engineering
Islamic University of Technology.
A Subsidiary Organ of the Organization of Islamic Cooperation.
Dhaka, Bangladesh.

May 2022

Declaration of Authorship

I, Md. Hamjajul Ashmafee, declare that this thesis titled, 'Development Of An Explainable Natural Language Query Driven Data Visualization System' and the work presented in it is my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Any part of this thesis has not been submitted for any other degree or qualification at this University or any other institution.
- Where I have consulted the published work of others, this is always clearly attributed.

Submitted By:

(Signature of the Candidate)
Md. Hamjajul Ashmafee- 171041012
May 2022

Development Of An Explainable Natural Language Query Driven Data Visualization System

Approved By:

Dr. Abu Raihan Mostafa Kamal
Thesis Supervisor and Professor,
Department of Computer Science and Engineering,
Islamic University of Technology.

Dr. Md. Hasanul Kabir
Professor,
Department of Computer Science and Engineering,
Islamic University of Technology.

Dr. Md. Moniruzzaman
Assistant Professor,
Department of Computer Science and Engineering,
Islamic University of Technology.

Dr. Chowdhury Farhan Ahmed
Professor,
Department of Computer Science and Engineering,
University of Dhaka, Dhaka, Bangladesh.

Abstract

Nowadays, visual interactive systems (Vis) are attracting more attention in research and industries because of their effectiveness in conveying information. Additionally, to make rational decisions based on extracted data, Vis is critical for identifying and comprehending trends, outliers, and patterns in data. Existing research has employed a broad range of methodologies to yield visualization insights into certain decision-making systems, allowing participants to perceive a specific problem from a wide range of viewpoints. However, there are still enough scopes to design a new Vis especially using visualization-oriented natural language interface (V-NLI) where state-of-the-art NLP techniques are utilized to visualize the data according to the user's NL queries. Furthermore, in several real-life decision-making scenarios, this DV tools are required with proper explanations to build trust on the predictions of the model. In this regard, we propose a framework for explainable V-NLI based data visualization system. Therefore, (i) we developed a deep learning-based NLP framework to extract key information to generate proper visualization type (viz-type) on given user query. (ii) Next, we extend our prior model to an explainable visualization model that not only accurately visualizes the desired data but also explains why it appears depending on the given natural language query (NLQ).

Keyword - Data visualization; V-NLI; LSTM; XAI, LIME

Acknowledgements

I would like to express my whole-hearted gratitude to Allah Subhanu Wata'ala for giving me strength to complete this study, and being with me when none was there beside me. I would also like to express my grateful appreciation to Dr. Abu Raihan Mostafa Kamal, Dr. Md. Azam Hossain and Dr. Md. Rafiqul Islam for their constant motivation and support throughout this study.

This work is partially supported by

Network and Data Analysis Group (NDAG)
Department of Computer Science and Engineering (CSE)
Islamic University of Technology (IUT)

Contents

List of Figures

List of Tables

Dedicated to my parents and siblings for their lifelong dedicated support to my education ...

Chapter 1

Introduction

In today's world, organizing, presenting, and highlighting data in a meaningful way is a significant contribution to the data mining sector [6, 7]. Despite the fact that there are numerous methods for managing, displaying, and recognizing data to users, data visualization (DV) can play an important role in representing data [8, 9]. However, to interact with these DV tools, it requires users to learn how to transform their task intention into tool specific operations [10]. Thus, it is required to provide a user friendly graphical representation, where natural language interfaces (NLIs) may be considered as a promising way to interact with such visualization systems for analyzing data [5, 11, 12]. From the key existing studies as shown in Table 1.1, it is observed that NLIs are gaining popularity day by day because they help to improve the usability of visualization systems in interactions, feeding queries, providing commands, and eliminating complexities [13, 14]. Moreover, domain experts only know what questions they could ask through NLIs, but they do not have enough technical depth to manage a visualization system for investigating their inquiries [15].

In the past few decades, several researchers have shown an interest in exploring natural language query (NLQ) processing in the data visualization domain [8, 17]. For example, Cox et al. [18] presented an initial prototype of an NLI that supported well-structured commands to specify visualizations of large and complex data. Liu et al. [3] proposed a pipeline based on a deep learning model to generate visualizations answering a natural language question related to the aggregation functions on tabular data. However, the existing NLQ-based visualization system have several difficulties, such as: (i) implementing free-form natural language

Table 1.1: Existing key studies: explainable natural language query-based visualization interactive system

Reference	Contributions	Approaches
Narechania et al. [5]	They developed a NLI for visualization based on the input of a tabular dataset and a NLQ about that dataset to flexibly specify and interact with visualizations.	(i) Low level implementation of NLP for natural language interface. (ii) JSON objects for analytical specifications. (iii) Vega-Lite specifications for visualizations.
Chowdhury et al. [6]	They recommended a visual interaction with a support of NLP to make interaction easier for complex data and remove modality limitations.	(i) A finite state model to combine input as speech and direct manipulations made by the user. (ii) Multiple coordinated views for data representation. (iii) A frame-based dialog management tool to detect user's intents and slots, and manage the context
Kim et al. [15]	They suggested an automatic chart question answering pipeline to analyze chart data, answer the questions, and explain those answers to build the trust and transparency.	(i) Vega-Lite chart for data extraction and visual encoding. (ii) NLP techniques to analyze the given NL questions about the chart. (iii) Sempre ML model to answer that question about the chart. (iv) A template-based NL generation approach to generate the explanation.
Liu et al. [3]	They contributed an automatic pipeline to generate visualizations with annotations to answer NL questions about tabular data given as input.	(i) Pre-trained BERT language model to vectorized input NL questions and table headers. (ii) A deep neural network to extract related data areas and corresponding aggregating types and tasks.
Fu et al. [1]	They have developed a new dataset, Quda, to provide sufficient training samples and implement visualization oriented NLI in free-form natural language. This dataset contains more than fourteen thousand queries with ten analytic tasks for visualization.	(i) Collect seed queries from the target users. (ii) Use the crowd to generate and validate paraphrases.
Kumar et al. [16]	They built a system to detect sarcasm in dialogues and conversational threads on social media extracting key features and trained that ML model to detect sarcasm in real time.	(i) XGBoost, an ensemble supervised ML algorithm trained with MUStARD dataset, to predict sarcasm (ii) two post-hoc interpretability approaches, LIME and SHAP to generate the explanation.

input due to the complexity of human language grammar, (ii) deficiency of sufficient NLI datasets aimed at data visualization for training a more generalized ML model and so on. Therefore, to address these issues, existing research has proposed several visualization systems following different strategies [8]. For example, DataTone [19] uses a combination of lexical, constituency, and dependency parsing to let people specify visualizations through NL as well as mixed-initiative interaction to resolve these ambiguities through GUI widgets such as drop-down menus, lists, thumbnails, and others. Articulate [20] is a natural language conversation-based interface that accepts instructions from users and generates graphs based on the categorization of visual analytic tasks, relieving the user of the burden of learning a complicated user interface in order to create a visualization. FlowSense [21] makes use of cutting-edge natural language processing methods to aid in the creation of dataflow diagrams, allowing flexible visual data exploration and the development of a dataflow diagram demonstrating the system's functioning. Eviza [22] and Evizeon [23] allow both specifying and interacting with visualizations through a natural language interface to analyze data. However, based on the current research, it has been shown that there are still certain constraints that limit the efficacy of these DV tools. Mostly rule-based approaches to NLQ analysis that rely on domain experts to construct visualization-oriented language rules [24]. Moreover, large-scale visual data analytics training corpora are required for the development and benchmarking of learning-based approaches for the implementation of visualization-oriented NLIs (V-NLI). [1].

On the other hand, these advanced ML techniques for DV need to be "explainable" in order to improve the system and its performance. Because it is still hard to figure out what really happened in these systems, and the visualizations of complicated data or queries may need to be checked by humans. When a visualization model is very complicated, it may be hard to explain how the results are made, which could make some users doubt the prediction. Also, if the model is easy to understand, it will quickly gain the trust of its users. Also, model developers may be able to learn about important features so they can make a good decision [15, 25, 26].

In recent years, the accuracy of the visualization models for figuring out the inner meaning of the context and the key features that are needed for free-form natural language queries have become more important. A lot of popular NLP approaches based on machine learning and deep learning have been made, but they should be reconfigured with modern DV tools to make them more accurate [27]. In some cases, a high prediction rate is needed to make a specific decision in regulatory bodies where understanding and accountability of the model's explanation are very important. Some NLP models can't take into account all of the possible factors that could affect the overall decision. In these cases, explainability can help find the features that affect both local and global decision making so that the underlying model can be changed. This lets us know what will happen if a parameter or input is changed. [16].

1.1 Problem Statement

Data visualization is commonly employed to facilitate the visual exploration of vast amounts of data and to extract the key insights inherent within it. V-NLIs have increased the effectiveness and usefulness of rendering visuals in response to NL queries from the user. Based on recent research, we've found the following problems that still need to be fixed:

1. Typically, data visualization systems that support V-NLIs employ rule-based techniques that provide less room for template-free instructions. For instance, if the template is built to search the maximum like "*Find the maximum of...*", the user's utterance such as " *What is the finest...*" might not be appropriately interpreted due to the rigidity of the templates. Even if they are semantically identical, the underlying regulation structures may prevent the extraction of accurate information. As the majority of rules are developed by hand, morphological and syntactical analysis in this area demands a profound comprehension of language. Also, domain specialists in data analytics need have a basic level of language comprehension to communicate with these DV tools. If the rules are complicated, the system will be difficult to comprehend quickly and will prevent users from freely exploring data. [5, 28].

2. Building a strong dataset is necessary for training a machine learning (ML) model, which may be used to find out the underlying pattern while relieving domain experts of the responsibility of designing complicated rules. This problem is particularly challenging in the field of natural language processing (NLP), since the semantic feature space is constantly increasing [29]. Additionally, as the datasets in this area are exceedingly limited, it is difficult to create a model for data visualization in this domain [1]. Another key problem in this discipline is preprocessing the dataset in order for the model to learn how to extract the relevant information from a given query [1, 24].

3. Another problem in the field of AI is that the systems we make based on DV oriented ML, are getting harder for users to understand. So, users in the governmental chairs are less likely to use these kinds of models because they are less reliable in some sensitive and important situations [16]. Deep learning techniques have been used to do state-of-the-art work in a number of natural language processing (NLP) tasks. Even though these deep neural network models perform very well, it is hard to get people to believe their predictions because they lack the insight and explainability. This prediction related explanation is required to understand how they are made in a dynamic environment [30].

In order to overcome the aforementioned limitations, we must design a pipeline that extracts key information about data visualization from a given natural language query using a deep learning-based natural language processing approach. The explainable module, which will generate the interpretation of the prediction generated by the prior deep learning model, will be added later to this pipeline, which will require more development.

1.2 Research Challenges

Despite the fact that the research area encompasses several state-of-the-art natural language processing (NLP), data visualization, and explanation generating techniques and models, there are still some difficulties to overcome in order to solve the limitations listed in the section 1.1:

- **Dataset Collection and Preprocessing:** This is difficult to do for two reasons: first, the NLIs should accurately represent how professional data analysts interact with data when performing various activities on different data sets for visualization; and second, the collection of such a corpus is time-consuming. Additionally, despite the fact that we are extremely capable of collecting data from social media users, V-NLIs are not widely used in this domain [1, 31]. Through the use of the crowd force, it is possible to validate high quality and huge volume queries derived from seed queries as well as enriched paraphrase queries, which is a very difficult process [1]. Also, as was already said, having a person evaluate low-quality semantic queries takes time and costs money.

- **Development a Learning-based Approach:** Currently, the semantic parsing phase of natural language in existing V-NLI models is based on simple word or template matching, which is unable to deal with the problems of ambiguity and underspecification that naturally occur in natural language. In order to improve accuracy, we should eliminate the usage of pre-defined templates when creating a visualization, as this technique still needs users to be familiar about visualization design principles. A V-NLI model that does not rely exclusively on the words or template must be developed in order for a user to be able to generate an open domain question without relying on the words or template. Because of the reduced barrier to entry in the open domain questions, visualization may be viewed by a large number of individuals if the open domain questions can be parsed accurately [3].
- **Explanation of AI System's Decision:** Current Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) systems fail spectacularly when it comes to establishing confidence in the minds of users while making decisions. In order to make a decision, it is quite difficult to interpret the information generated by the employed NLP techniques [32]. The research should also be comprehensive in order to identify the features that contribute to the decision-making process based on their saliency score and to provide an effective and intelligible explanation to the user in order to gain trust [16].

1.3 Research Contribution

To build trust during decision making, this study tackles challenges in NLP techniques for extracting key information about data visualization from a given V-NLI and making this framework explainable to users. A deep learning-based NLP model is suggested in the first section of the study to extract key information about data visualization. The second half of this study proposes another explainable AI-based framework to make the prior model understandable to the user. This thesis' main contributions are summarized as follows:

- First, we have looked at the existing dataset on V-NLIs, which contains a large size corpus of seed inquiries from domain experts in the fields of data analysis and data visualization. Validating these queries in order to retain their quality is conducted as part of the data preparation to be tailored for a learning-based model.

- Second, we have created a natural language processing model designed to extract crucial information from the user-supplied NLQ for data visualizations. To construct a natural language query-based data visualization system with an emphasis on deep learning, we examined several ways to maintain improved accuracy and validation.
- Third, to make our deep learning-based DV-oriented NLP technique comprehensible so that users can readily develop trust in this model, we have addressed a number of challenges, such as exploring post-hoc identifying contributing features, generating surrogate models, and so on.

1.4 Thesis Outline

The rest of the dissertation is organized as follows. Chapter 2 deals with the necessary background and literature review for this study. In Chapter 3, the proposed methodology regarding developing the deep learning based NLP approach for key information extraction and the explainability model for explanation have been discussed in detail. The experimental designs for deep learning based NLP approach and explainability model are presented separately in Chapter 4. Chapter 5 discusses the evaluation metrics, results of NLP based classification approaches in ML and DL domain, process to generate the explanation for ML model and the necessary aspects to consider while conducting additional research in this domain. Chapter 6 draws a conclusion to the current study and discusses future directions. The final segment of this study contains all the references.

Chapter 2

Literature review

In the booming field of data analytics, the use of interactive visualization is gaining popularity due to its ability to visually emphasize key information for consumers. The visualization-oriented natural language interface (V-NLI) has increased its efficacy by allowing for the compilation of visualizations based on the user's natural interactions. Additionally, it assists the user in focusing on the task at hand rather than the tool's interface. The fast advancement of NLP technology has created an excellent opportunity to further develop this discipline. On the other hand, explainable artificial intelligence (XAI) has gained traction in industry and research as a means of bridging the divide between why and how an AI system makes a certain judgement. V-NLI-based data visualizaion (DV) is not unrelated to this interest, since it is utilized to make governing decisions in a variety of governance roles.

Through the perspective of literature, we will gradually study data visualizations, V-NLIs, advances in NLP, and the evolution of XAI models in this chapter.

2.1 Data Visualization: Toolkits, Grammars and Visual Encoding

There are several toolkits in visualization research that are designed to help the formulation and development of visualization simpler. Furthermore, a number of visualization creation tools are offered to aid visual exploration of tabular data. Users of the methodologies and tools discussed above must establish the link between data properties and visual channels, which may make visual data exploration difficult for novices. As a result, several works also provide recommendations for suitable visualizations. Based on established rules, *Keshif* [33], *SeeDB* [34], and *Voyager* [35] focus on the appropriate data attributes and offer a selection of charts. For data management, visual encoding, and interaction, these visualization tools employ a set of configurable operators. Despite the fact that new combinations of operators can provide customized views, we've found that most unique visualizations require programmers to change the visualization through scripting. As a result, these visualization frameworks function well when the designers of the visualizations are software engineers, but they are prohibitive for more broad audiences such as web designers.

Many business tools, such as *Tableau* [36], *PowerBI* [37], and *ManyEyes* [38], allow us to create visualizations without having to write any code. *Polaris* [39], *Lyra* [40], and *iVisdesigner* [41] allow users to drag and drop attributes from a table, define encoding techniques, and then construct visualizations using the templates with the specified encodings. The visualization tools are primarily designed to specify and render visuals in a structured manner. Such toolkits as *D3* [42], *Vega* [12] and *Vega-Lite* [2] facilitate visualization design using declarative specifications, allowing for quick visualization design and prototyping. The toolkits assist the user in developing visualization specifications programmatically. The researchers are also working on visualization grammars that give a high-level abstraction for creating visualizations, reducing the need for software engineering expertise.

The visualization tools create the fundamental plots by mapping data attributes to visual channels like location, color, shape, and size, and may incorporate standard data transformations like binning, aggregating, sorting, and filtering [2]. They may employ a portable file format (for example, JSON or XML) to exchange simple visualization grammar and build appropriate visualizations from a range of programming languages.

2.1.1 Data Visualization: How It Works

These tools are composed of visualization grammar, which enables the creation of customized visualizations. Sometimes, they give a higher-level language for abstraction and visualization development. *Grammar of Graphics* [43], *ggplot2* [44], and *ggvis* [45] are often used examples.

We describe how *Vega-Lite* [2] generates a visualization plot in this section. It begins with the simplest Vega-Lite definition, also known as a unit specification, which provides a single Cartesian plot with a specified mark type for data encoding. It defines such a plot using a specified dataset, a certain mark type (e.g. bars, lines, symbols), and a set of encoding definitions such as size, color, and x-y coordinates.

$$unit := (data, transforms, mark - type, encodings) \tag{2.1}$$

$$encoding := (channel, field, data - type, value, functions, scale, guide) \tag{2.2}$$

They integrate several composite operations using numerous unit plots to create an algebra for generating composite views. Each operator is responsible for integrating or aligning any necessary underlying scales and axes. The composite view is generated using supporting operators such as layer, concat, facet, and repeat. The layer operator receives numerous unit parameters and generates a view in which succeeding charts are superimposed on one another. *Vega-Lite* includes horizontal and vertical concatenation operators for concatenating views. In Figure 2.1, it shows an example visualization in *Vega-Lite* with $mark - type = line$ and its visual definition is instantiated in a JSON format.

```
{
  "data": {
    "url": "data/weather.csv",
    "formatType": "csv" },
  "mark": "line",
  "encoding": {
    "x": {
      "field": "date",
      "type": "temporal",
      "timeUnit": "month" },
    "y": {
      "field": "temp_max",
      "type": "quantitative",
      "aggregate": "mean" },
    "color": {
      "field": "location",
      "type": "nominal" }
  }
}
```

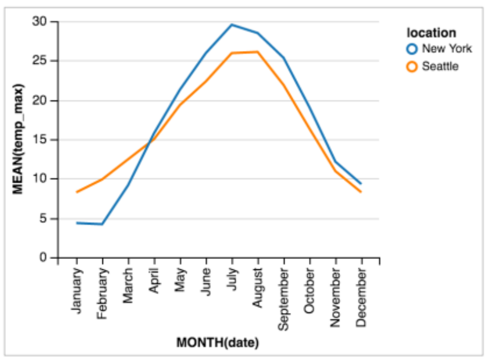


Figure 2.1: A line-chart from Vega-Lite [2]

2.1.2 NLI for Data visualization: An interface for visualization

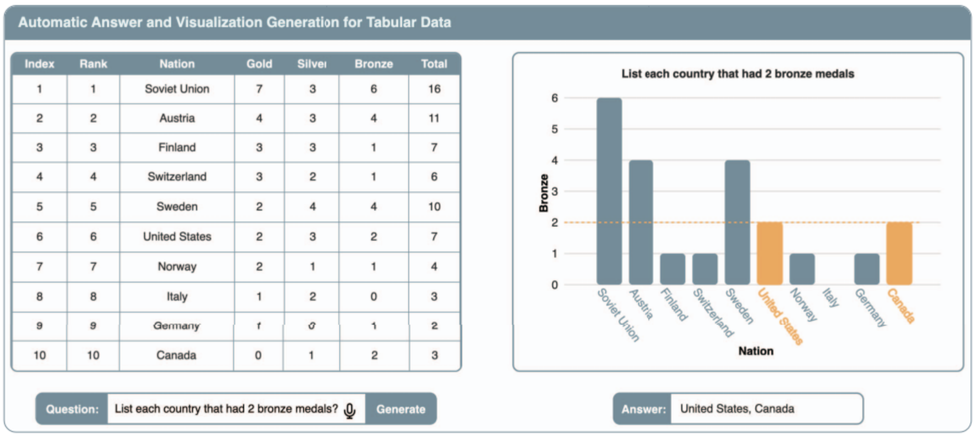
In 2001, Cox et al. [46] introduced an initial version of a NLI that allowed for the specification of visualizations using well-structured commands. The fast evolution of Natural Language Processing (NLP) technology over the years has created an excellent opportunity to investigate a natural language-based interaction paradigm for data visualization [24]. With the aid of powerful natural language processing toolkits [5], a wave of visualization-oriented natural language interfaces (V-NLI) has evolved as a supplement to traditional tool-based interaction. It enables the generation of visualizations based on the user's NL queries and improves the system's usability and efficacy for beginner users. Later, Articulate [20] offered a two-step approach for creating visualizations from natural language queries that automatically extracts analytic task and data properties from the user query and then provides the relevant visuals based on that information. It uses keyword-based classification methods to determine the weighting of a user's ambiguous query in each feature. Sisl [47] is a multi-modal interface that supports a variety of input modalities, including point-and-click, textual NL input, and NL input through speech. Analyza [48] is a data exploration system that combines V-NLI and a structured interface.

Microsoft Power BI [37], IBM Watson Analytics [49], Wolfram Alpha [50], Tableau [36], and ThoughtSpot [51] all include V-NLIs to improve the analytic experience for beginner users by including features such as autocompletion, inference for unspecified utterances, and so on. DataTone [19] pioneered the use of ambiguity widgets to manage query ambiguities using a mixed initiative method. Beyond data exploration, FlowSense [21] added semantic parsing to a dataflow-based visualization system using V-NLI. This enables users to interact with multiple views inside the DV system. Eviza [22] and Evizeon [52] investigated how to build and manipulate visuals using analytic conversations.

With the advancement of hardware devices, interest in synergistic multimodal visualization interfaces has increased significantly. Orko [53] was the first solution to integrate touch and spoken input on tablet devices, whereas Data@Hand [54] focuses on smartphones and provides additional interactions for visual exploration and analysis of graph data. Later, InChorus [55] included a pen as a third mode of interaction for a more uniform interaction experience.

The techniques are mostly implemented using rule-based language parsers that only handle limited free-form NL input. Rather than creating visuals, Text-to-Viz [56] intends to create infographics from natural language statements by collecting 800 valid proportion-related phrases and using a machine learning model to analyze utterances. Although promising, Text-to-Visualization does not enable a wide variety of analytic tasks. A sample V-NLI is depicted in Figure .

V-NLIs' performance and application scenarios are largely dependent on language parsers. The most advanced NLP models have achieved near-human accuracy in a variety of tasks, including semantic parsing, text categorization, and paraphrase synthesis. However, few have been applied to V-NLIs with a focus on visualization.



Index	Rank	Nation	Gold	Silver	Bronze	Total
1	1	Soviet Union	7	3	6	16
2	2	Austria	4	3	4	11
3	3	Finland	3	3	1	7
4	4	Switzerland	3	2	1	6
5	5	Sweden	2	4	4	10
6	6	United States	2	3	2	7
7	7	Norway	2	1	1	4
8	8	Italy	1	2	0	3
9	9	Germany	1	0	1	2
10	10	Canada	0	1	2	3

Question: List each country that had 2 bronze medals?

Answer: United States, Canada

Figure 2.2: Interface of *ADVISor* [3]

2.2 Natural Language Processing Toolkits and Approaches

Natural Language Processing (NLP) is a computer-assisted analytic approach aiming at automatically analyzing and interpreting human language. It facilitates the extraction of valuable insights from textual information without requiring tedious computational effort. This discipline has gained significant traction in recent years and is now one of the most dominating in data science. NLP is an area of artificial intelligence (AI) that enables machines to automatically execute repetitive activities such as summarization, machine translation, pattern matching, sentiment analysis, speech recognition, and many more [57]. It has not only brought about revolutionary changes in people's everyday lives, but also in business practices. With the advent of AI bots in smart devices such as Alexa, Cortana, Siri, and Google Assistant, the use of natural language processing has increased exponentially. NLP has seen improvements in precision, speed, and even methodologies that are now widely employed to handle complicated issues [58].

2.2.1 Natural Language Processing Toolkits

Natural language processing assists us in comprehending text and gaining crucial insights. The use of NLP technologies improves our comprehension of how language may function in specific situations. There are currently an abundance of natural language processing tools on the market. Primarily, they are employed in a variety of NLP-related activities, including text categorization, part-of-speech tagging, entity extraction, tokenization, parsing, stemming, semantic reasoning, dependency parsing, removing stop-words, generating N-grams and many others [5]. Recent research has uncovered a large number of NLP toolkits, the most notable of which are NLTK [59], Stanford CoreNLP [60], Apache OpenNLP [61], SpaCy [13], AllenNLPSpaCy [62], GenSim [63], TextBlob Library [64], Intel NLP Architect [65], GoogleNLP [66], Flair [67], and Stanza [68]. These toolkits are extremely efficient and useful for assisting developers with NLP tasks. The most popular NLTK interface comprises text corpora and lexical resources and enables rapid integration of NLP services into systems by developers. Tf-idf and BoW are most common algorithms in these toolkits used to extract features from a given text. However, because these are general-purpose toolkits, domain-expert developers must learn how to use the toolkit as well as the underlying NLP principles. Knowing which dependency paths to traverse while processing queries or comprehending semantic similarity metrics are examples of these notions. The semantic parser can conduct a series of NLP sub-tasks on the query string of V-NLI to extract valuable details that can be used to detect relevant phrases. Internally, NL4DV [me/10] leverages NLTK and SpaCy to provide NLP-related APIs that encapsulate and mask the NLP implementation specifics. As semantic and syntactic analysis is a fundamental task for V-NLI, the vast majority of NLP-based systems, along with FlowSense [69] offer semantic parsing by directly employing NLP toolkits. Moreover, in order to construct visualization systems, developers must write extra code to translate the output of NLP toolkits into visualization-relevant concepts (e.g., characteristics and values for applying data filters), which may be a difficult and time-consuming process [70].

2.2.2 Rule based Natural Language Processing Approaches

Many researchers used the rule-based method and relied on hand-constructed language rules in constructing their systems in the early days of NLP because of a lack of resources and their constraints. It usually consists of morphological and syntactic analyzers that are utilized for certain tasks. Rule-based transformation systems and tools are built on a solid foundation of language understanding. NLP analysts use a knowledge-based technique to analyze language behaviors that occur inside text employing syntactic, semantic, and discourse information in the development of these rules to extract the coded segments [me/41], [me/64], [me/62].

Flowsense [57/280], for example, is a dataflow visualization system that uses the Stanford SEMPRE framework [57/180], [57/288], to map natural language utterances to appropriate tags and placeholders via intermediary logical forms. NL4DV [me/10] keeps track of aliases and lets developers customize them. It checks for syntactic and semantic similarities between N-grams and a lexicon of data attributes, aliases, and values as it iterates over the created N-grams. NL4DV uses the cosine similarity function for syntactic similarity and the Wu-Palmer similarity score based on WordNet [57/163] for semantic similarity. Analyza [57/52] extends the lexicon with a proprietary knowledge graph and uses additional heuristics to infer information from data properties.

For complex rules, rule-based systems quickly become unmanageable. The interrelation between these rules gets excessively complicated with little over a hundred rules. The sheer volume of data and rules that may be applied quickly proved these techniques ineffective. Recent techniques, on the other hand, incorporate methods that take advantage of the massive amount of data available for training language models. To put it another way, contemporary methods to language processing make use of data-driven methodologies to achieve the aims of language comprehension. [me/67]

2.2.3 ML based Natural Language Processing Approaches

Unlike the rule-based method, the ML technique uses cognitive modules that can learn from historical data to automatically infer rules. ML algorithms are now being used to carry out numerous NLP jobs. These algorithms' parameters are historical data from which features are synthesized, and this features-based data is then utilized for prediction or classification [me/41]. In NLP, supervised learning is the most often used approach for solving problems. The essential concept underlying supervised machine learning models is that they infer rules from training data automatically. Hidden Markov models (HMM), conditional random fields (CRF), maximum entropy (MaxEnt), support vector machines (SVM), decision trees (DT), Nave Bays, and deep learning are the most often used machine learning models for ambiguity resolution [me/64]. These approaches are used in a lot of current NLP toolkits as well. Based on statistical and semantic aspects in the textual data, machine learning algorithms are employed to discover the complex patterns within the text. Because it does not require an NLP expert to develop the rules, using machine learning to infer underlying rules can be more cost-effective than rule-based techniques (which must not to say that expertise is not required at all).

Text-to-Viz [13] is an application for creating infographics using natural language sentences with proportional data. The doctors collected 800 genuine proportion-related statements and used machine learning to parse them. Text-to-Viz, while promising, does not provide queries for a wide range of analytic tasks. VizML [31/86] is a DV technique based on machine learning that may be used to improve visual mapping.

The machine-learning approach's success, on the other hand, is greatly dependent on having a large number of training instances from which to learn and being able to identify a relevant semantic feature space [me/66]. However, few have been applied to visualization-oriented V-NLIs. [me/31].

2.2.4 DL based Natural Language Processing Approaches

Deep learning (DL) is gaining popularity in recent years because to its accuracy when trained with vast amounts of data. It is one of the machine learning classes that performs much better on unstructured data [me/66, 63]. The text is seen as words in sequence by RNNs, which are designed to capture text structures and dependencies on words in order to categorize the text. When compared to the Feed forward model, the vanilla RNN model performed poorly. Though there are many different types of RNNs, the most common designs are Long Short-Term Memory (LSTM), GRU, and Bidirectional RNN, which are used to simulate the dependencies [me/66]. By solving the gradient vanishing concerns that vanilla RNNs have , LSTM incorporated a memory cell for remembering values in random time intervals and gates for regulating information flows inside and outside of each cell [me/63]. Transformers, BERT, seq2seq, and GPT-2 models have recently been popular for generating predictions for class surfaces (label mask) based on context [me/65]. However, owing of the lack of high-quality textual datasets to aid training and evaluation, V-NLI implementations are limited.

Liu et al. [12/15] proposed a method to automatically extract features from visualization charts and create explanations in plain language based on a deep learning model to improve the user's understanding of data characteristics in visualizations. Fu et al. [57/62] used an advanced pre-trained model, BERT [57/51], to train a multi-label task classification model based on a dataset of NL queries for visual data analytics. It use a deep neural network to construct the embeddings of the NL query and table headers, which are subsequently utilized by a deep neural network model to determine essential data visualization properties.

2.2.5 RNN and LSTM

In reality, we must depend on previous experiences rather than constructing every events from scratch. A typical neural network, however, cannot learn from prior occurrences since knowledge from one step does not impact the next. In contrast, RNN acquires information from the preceding phase.

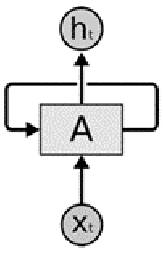


Figure 2.3: A single RNN architecture

the above diagram in Figure 2.3, a portion of a neural network evaluates the input $x(t)$ and returns the result $h(t)$. A loop permits the transmission of data from one network node to the next. These loops make recurrent neural networks difficult to perceive at first. but, It turns out that they are not very different from a typical neural network.A recurrent neural network is made up of many copies of the same network. Let's consider the unrolling of the loops in Figure 2.4

In this way, recurrent neural networks are similar to sequences and lists. They are the best neural network architecture for this data. These networks are made up of repeating neural network modules. In typical RNNs, this repeating module will have a simple structure, like the single *tanh* layer in Figure 2.5.

Recurrent Neural Networks exhibit poor short-term memory. If a sequence is sufficiently lengthy, it will be difficult to transmit information from previous time

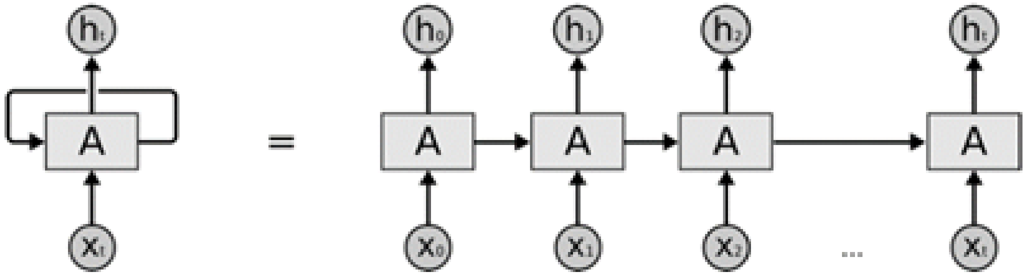


Figure 2.4: Unfolding a single RNN in different timestamp

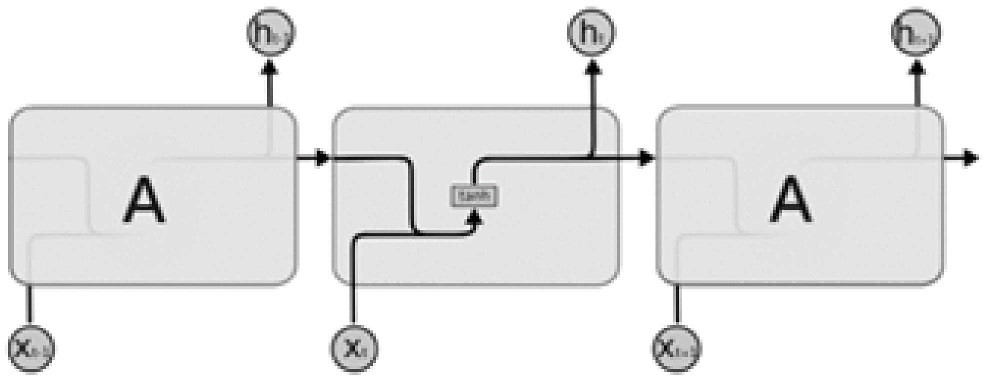


Figure 2.5: *tanh* activation function in RNN

steps to newer ones. During back propagation, recurrent neural networks experience the vanishing gradient issue for the *tanh* activation function. Gradients are values used to adjust the weight of a neural network depending on the activation function.

The vanishing gradient issue occurs when the gradient decreases as it back propagates in time. When a gradient value gets incredibly small, it contributes little to no learning. Therefore, if we attempt to analyse a lengthy text in order to make predictions, RNNs may omit crucial information from the beginning.

LSTM is a specific kind of RNN. A Long Short-Term Memory (LSTM) recurrent neural network architecture was introduced by Hochreiter et al. [15]. This model can solve the issue of vanishing gradients that occurs in prior models. They introduced a memory cell unit to release the ordinary node. The memory cell unit makes it possible that the gradient can flow through many time steps without vanishing or exploding by a self-connected recurrent edge [16]. Input gate and output gate have learnable parameters to regulate the amount of information that can access to the cell. Forget gate was introduced into LSTM by Gers et al. [17] which gives the ability to learn to forget internal resources of a memory cell unit to avoid the network break down caused by the state grow indefinitely. We use $X = (x_1, x_2, ..., x_N)$ to denote the input sequence, where $x_t \in R_l$. There are many variants of the LSTM model, here we show one adopted by Rocktaschel[8] as follows in Figure 2.6.

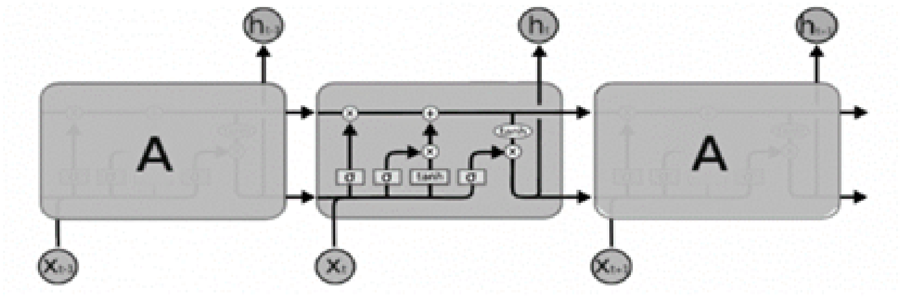


Figure 2.6: Unfold version of LSTM network in different timestamp

This LSTM offers a novel structure in place of conventional RNN which is called a memory cell. The main components of a memory cell are - input gate, a neuron with a self-recurrent connection, forget gate, and output gate as shown in Figure 2.7.

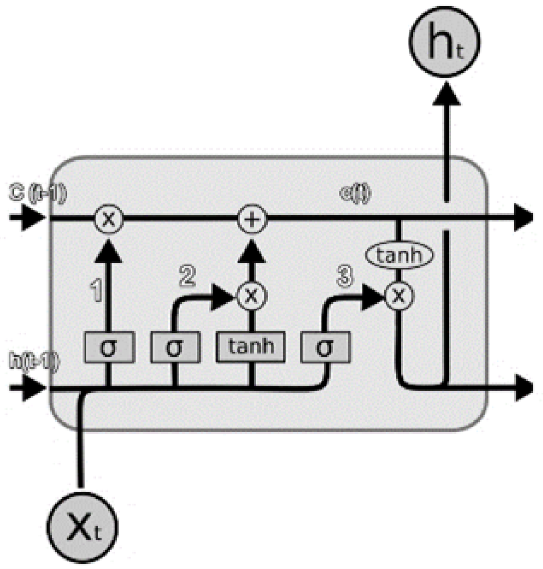


Figure 2.7: A single LSTM architecture

The core part of the LSTM is the cell state which works straightly with minor interaction throughout the chain. It allows the information to be flowed without any change. The LSTM cannot delete or add information to the cell state. However, it features structures known as gates that allow information to pass through if desired. The gates consist of a *sigmoid* neural network layer and a pointwise multiplication process.

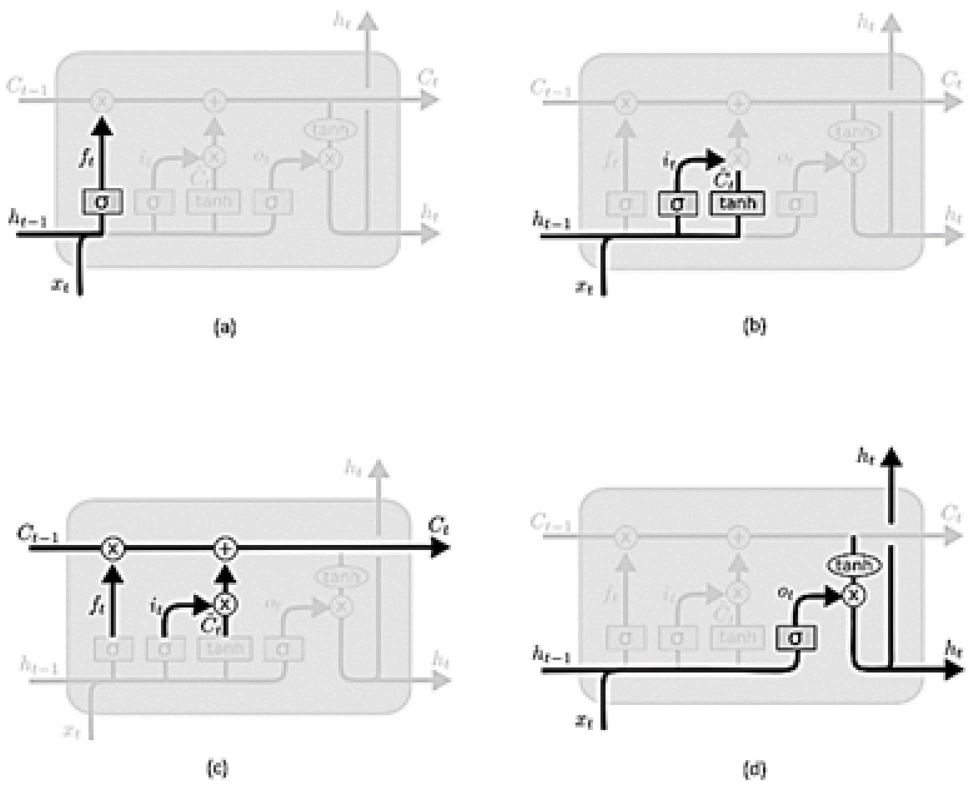


Figure 2.8: Different gates in a single LSTM architecture

The first step in our LSTM is to choose which cell state information will be discarded. This choice is done via a sigmoid layer known as the ”forget gate layer.” It examins at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. $A = 1$ represents “completely keep this” while $A = 0$ represents “completely get rid of this.” It works as the following equation in Figure 2.8(a).

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.3}$$

The next step is to determine what additional information will be stored in the cell state. This consists of two parts. First, a *sigmoid* layer known as the “input gate layer” decides which values will be updated. Next, a *tanh* layer creates a vector of new candidate values, C_t, that could be added to the state. In the next step, we'll combine these two to create an update to the state. It works as the following equation in Figure 2.8(b).

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.4}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.5}$$

It's now time to update the old cell state, C_{t-1}, into the new cell state C_t. The previous steps already decided what to do, we just need to do it. We multiply the old state by f_t, forgetting the things we decided to forget earlier. Then we add $i_t * \tilde{C}_t$. This is the new candidate values, scaled by how much we decided to update each state value. It works as the following equation in Figure 2.8(c).

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.6}$$

Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version. First, we run a *sigmoid* layer which determines what parts of the cell state we're going to output. Then, we put the cell state through *tanh* (to push the values to be between -1 and 1) and multiply it by the output of the *sigmoid* gate, so that we only output the parts we decided to. It works as the equation following in Figure 2.8(d).

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.7}$$

$$h_t = o_t * tanh(C_t) \tag{2.8}$$

2.3 Explainable AI: Techniques and Tools

Many recent scientific and technical advances include machine learning. In many uses of machine learning, users must blindly trust a decision-making model. In contrast, in some delicate and crucial circumstances, the user may request explanation for the model against a particular conclusion. Before making judgments based on ML models in healthcare, government services, the financial system, national security, and the energy sector, a certain degree of assurance is required. Even though many machine learning models are black boxes, knowing the logic behind the model's predictions can help users decide whether to trust them as depicted in Figure 2.9.

In several natural language processing (NLP) tasks, deep learning techniques have yielded state-of-the-art outcomes. Although the performance of these deep neural

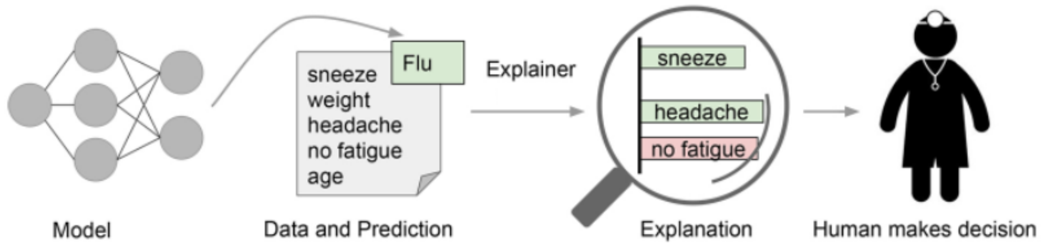


Figure 2.9: A general architecture of an explainable AI model

network models is outstanding, it is difficult to persuade individuals to believe their predictions because they lack the insight and explainability necessary to comprehend their dynamic decision-making process. People in the business are cautious to implement NLP-based strategies for decision making due to the fact that they are fundamentally black boxes for humans [me/70]. Nevertheless, eXplainable AI (XAI) tackles this challenge by enhancing the explainability and transparency of NLP models, hence removing the nature of a black box. A comprehensive overview of XAI principles, taxonomies, prospects, problems, and XAI tool adoption was presented by Danilevsky et al. [me/21].

To address this, model-agnostic post hoc explainability approaches such as LIME (73/Ribeiro et al., 2016) and SHAP (73/Lundberg & Lee, 2017) were created, in which an additional model is trained in order to comprehend, trust, and appropriately deal with the increasing number of AI applications [me/73]. Nurdin et al. [me/75] proposed a deep learning-based sentiment analysis model, which was further analyzed using various XAI methods such as LIME and SHAP, demonstrating that XAI is not restricted to delivering information on what occurs in the model, but may also help us comprehend and discern models' personalities and behaviors. Eval4NLP-2021 [me/74] presented a collaborative task on explainable NLP evaluation metrics with LIME and SHAP.

2.3.1 LIME: How it Works?

Local Interpretable Model-agnostic Explanations (LIME) is a tool for understanding and interpreting the underlying machine learning model while being model-agnostic. LIME was designed to approximate machine learning models using intelligible models. This is performed locally since it may be easier to grasp and approximate globally sophisticated machine learning models. [69/32]. The LIME descriptions will assist users to comprehend and interpret the model based on its contributing features. The LIME definition of explanations is as follows:

$$explanation(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g) \tag{2.9}$$

To be model-agnostic, an explainable model will learn the behavior of the underlying model by perturbing the input instance and seeing how predictions change. This is advantageous in terms of interpretability, since we may perturb the input instance by modifying components that make sense to humans (e.g., words or portions of an image), even if the model is employing far more complex components as features (e.g., word embeddings). Then, an explanation may be generated by approximating the underlying model as one that is interpretable. This explanation is derived from perturbations made to the original input instance by eliminating words or concealing images. The core concept underpinning LIME is approximating a black-box model locally using a simple model. Thinking locally is much simpler than attempting to simulate a global model.

In the latter stages of this method, a data set of perturbed instances is formed by "turning off" certain interpretable components. Observe the probability of an accurate model prediction for each instance that has been perturbed. On this data set, a simple (linear) model is trained that is locally weighted. Errors in perturbed situations are therefore taken into account because they are more crucial for making the right decision. Finally, the instance with the highest positive weights as an explanation is textually or graphically presented, discarding everything else.

This explanation is, intuitively, a local linear approximation of the model's behavior. Globally, the model may be rather complicated, but it is simpler to approximate it in the area of a specific instance. As an explanation, while considering the model as a black box, we perturb the instance we intend to explain and learn a sparse linear model around it. The Figure 2.10 explains the concept underlying this method.

The blue and pink background represents the the model's decision function, which is evidently nonlinear. The bold red cross represents the instance to be explained (let's name it X). We sample occurrences in the vicinity of X and weight them based on their proximity to X (weight here is indicated by size). Then, we develop a linear model (dashed line) that closely approximates the model around X, but not necessarily globally. Instead of training a global surrogate model to explain individual prediction, LIME trains local surrogate models. This entire procedure is described in the algorithm 1.

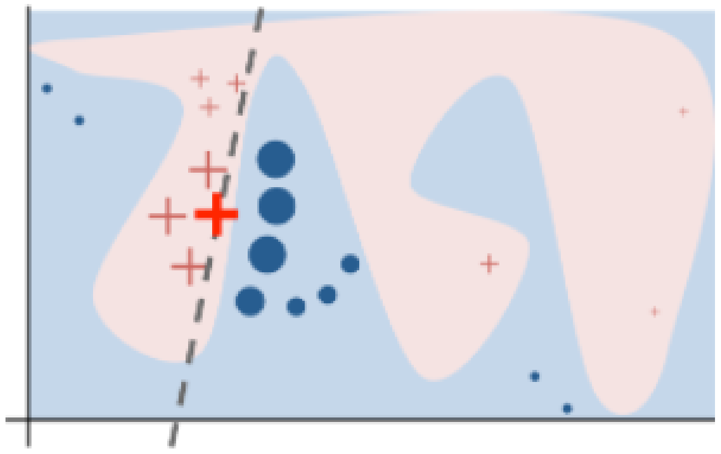


Figure 2.10: Intuition of LIME [4]

Algorithm 1 Explain a Prediction with a Trained Local Surrogate Model

```
1: Select the instance of interest for which the ML model is going to be explained.
2: Perturb new samples around the instance to get the predictions of the ML model.
3: Weight the new samples according to their proximity to the instance of interest.
4: Train an interpretable linear model on those samples with variations.
5: Explain the prediction of the instance by interpreting the linear model based on the
   contributing features.
```

Chapter 3

Proposed Methodology

In this chapter, we will construct a pipeline to extract critical information from a given query using a cutting-edge NLP technique. Through an intense training process, this learning-based model will comprehend the underlying structure of important information. In the end, this pipeline will be investigated further to transform this decision-making model from a black box to a white box.

3.1 Dataset Acuisation

3.1.1 QUDA [1]

We have used here the corpus, QUDA which has 14,035 diverse user queries annotated with basic low-level analytic tasks (e.g., correlate, filter, sort, find extremum and others). These analytic tasks are associated with the data tables to extract desired information to be visualized. This corpus consists of a number of queries annotated with high-quality features and large-volume queries designed in the context of V-NLIs. To accomplish this goal, the authors employed both expert and crowd intelligence. First, they collected the seed queries by interviewing data analysts which are treated as *"expert queries"*. Next, to diversify the ways of saying something, they employed the crowd to collect a number of sentential paraphrases for each query. Basically, these queries are restatements with approximately the same meaning. Finally, they applied a validation procedure for quality control. So, this dataset contains 920 utterances created by data analysts, and each are accompanied by 14 paraphrases on average. All queries were annotated with one of ten analytics tasks they have fixed earlier. They have characterized their data set in the following ways:

- **Abstraction: Concrete**
  Abstract queries are beneficial for generalizing an idea behind a specific activity, although their interpretation might vary. A query should offer enough information for V-NLIs to respond reasonably. So they concentrated on low-abstraction queries that specify tasks and values directly.
- **Composition: Low**
  A query's composition level defines how many sub-queries it contains. Based on these sub-queries, composition is a continuous scale from low to high. A high-composition inquiry has several sub-queries, and a V-NLI may answer it in stages. So, to make things easier, the searches in this dataset had a low composition level.
- **Type of Data: Table**
  The dataset influences the query syntax and semantics. For a network dataset, analysts could seek the shortest route between two nodes. However, in the tabular dataset, connections between objects are not expressly supported. Tables, trees, fields, clusters, sets, lists, and so on are all forms of data. This corpus targets queries from a tabular dataset.
- **Context Dependency: Independent**
  Because V-NLI is conversational, queries may depend on textual information. If a query is followed by another inquiry in the same context, the second query may look incomplete. Contextual queries are not considered in this dataset. This corpus' queries pertain to tasks or values connected with tasks.
- **Construction of QUDA**
  As a result, the created 36 real-world data tables encompassing 11 various areas such as health care, sports and entertainment. These 14035 queries cover 10 low-level analytic activities. They used expert and crowd intelligence to collect data in three steps. First, they interviewed 20 data analysts on writing queries for 36 data tables and 10 low-level jobs. This phase yielded 920 expert queries. Then they used a large-scale crowd-sourced experiment to gather sentential paraphrases for each expert inquiry. To assure data quality, they created a validation step incorporating both crowdsourcing and machine learning. The final step yields 13,115 paraphrases, with an average of 14 paraphrases per expert statement.

3.1.2 Dataset amendment for this research

As we target to develop a visualization system based on NLIs, the visualization types (viz_type) were not integrated with the corpus described above. So, we have used crowd intelligence to collect the corresponding viz_type against each of the queries from the 920 expert queries' set and share the results into the associative paraphrases. In this approach we have performed the following steps:

1. First, we described the task as *"Find the best visualization among the following figures that is perfect to present the information asked in the query."* to help crowd workers understand our task. The interface also encourages the crowd to select the appropriate viz_type that is suitable for the query asked to visualize data. We demonstrate both valid and invalid examples to explain our requirements. The interface shows a query and the corresponding visualization based on different visualization types (viz_type). The crowd was asked to select the best figure that is meaningful and perfect to understand the context from the displayed graph or chart against the given query. Finally, we asked employees an open-ended inquiry to get feedback and recommendations for this position. In the primary experiment, we built a mechanism to guarantee the quality of the visualization kinds obtained for each query. Before receiving an intensive fix, the subjects were told that the findings will be examined and verified.

2. In the validation step, we have calculated the normal distribution of the answers from the crowd for a particular query. After removing outliers outside of 95% data points, we considered the majority vote of a particular viz_type against an expert query. To satisfy the voting as majority, we have fixed its threshold value which should be maintained to be accepted (here it was 0.4 for each class).

3. Then we have copied this label of visualization types (viz_type) to the corresponding paraphrases of a particular expert query.

4. We have set up a web-based system to assist the participants in conducting comparison among the viz_types. The system displayed the viz_type descriptions, examples and requirements. We have delivered 400 queries for this survey along with ten gold standard instances for quality control (ten clear and unambiguous queries). We manually crafted this set of gold standard queries those are unambiguous and have a clear answer. We rejected any jobs where four of the gold tasks were failed to be selected.

3.1.3 Overview of the final dataset:

After preparing the dataset we have collected 10334 queries with their corresponding visualization types. From the collected data set we have got 4 different viz_type labels to cover all queries in the dataset. The distribution of the dataset is like the following in Figure 3.1:

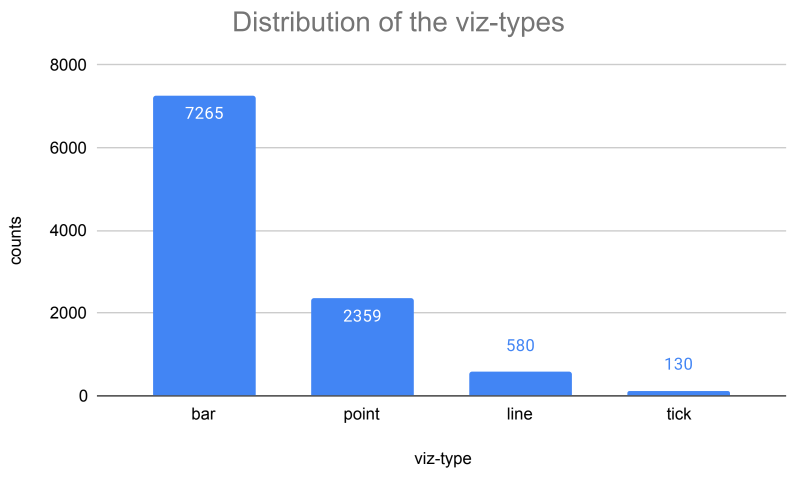


Figure 3.1: Primary distribution of QUDA [1]

Here, we can see that the number of samples of bar chart is very large as most of the queries in the dataset could be visualized with bar chart and histogram. Some others could be visualized with heatmap and scatter plot what is depicted with point. Line charts basically present the trend of the data and tick charts visualize the strip plot.

3.2 Dataset Preprocessing

3.2.1 Data augmentation to balance imbalanced dataset

The performance of the most machine learning and deep learning models in particular, depends on the quality, quantity and relevancy of training data. However, insufficient data in different class labels is one of the most common challenges in accurately implementing such models. This is because collecting of data covering all class labels is very costly and time-consuming in many cases. So here we can apply data augmentation to reduce reliance on data collection from the real world and build more accurate machine learning models faster.

Data augmentation is a collection of strategies for producing additional data points from existing data to artificially enhance the quantity of data. Making modest adjustments to data or utilizing machine learning models to produce additional data points are examples of this. By adding fresh and diverse instances to the dataset, it is possible to enhance the performance and results of machine learning models. In the NLP sector, it is becoming increasingly common to produce extra, synthetic data from existing data.

In the case with NLP, the data augmentation should be done carefully due to the grammatical structure of the text and should be used before training the model. The common approaches of the text data augmentation are as followings:

- **Back translation:** We convert the text data into another language and then back into the original language using this procedure. This may aid in the generation of textual data using various terms while maintaining the context of the text data. To do this, several language translation APIs are employed.
- **Synonym replacement:** Choose n words at random that are not stop words from the sentence. Replace each of these words with a random synonym for it.
- **Random Insertion:** Find a random synonym of a random word in the sentence that is not a stop word. Insert that synonym into a random position in the sentence.
- **Random swap:** Change the places of two words in the phrase at random.
- **Random Deletion:** Randomly remove each word in the sentence with probability p.

As our VNLIs are made of only one sentence providing a certain instruction based on the data tables to visualize the extracted data, we think each word in the query is equally important. In this case, back translation approach is not applied here as the APIs for more accurate augmentation is paid and has some limitations to paraphrase a large amount of data. Also, the random insertion, deletion and swap will change the instruction inherent to the query. So as a best and most effective practice, we have followed the synonym replacement via word embeddings [71]. We replace n number words with its synonyms (word embeddings that are close to those words) to obtain a sentence with the same meaning but with different words.

Here we have augmented the text by replacing with synonyms from the WordNet thesaurus [72]. WordNet is a lexical database for the English language which is a part of the NLTK corpus to find out the meanings of words, synonyms, and antonyms. Besides the neural network approach like word2vec, GloVe, this thesaurus can achieve similar objectives to find the synonym words [3]. To implement this augmentation, this step provides a substitution feature to replace the target word. Instead of finding synonyms purely, some preliminary checking makes sure that the target word can be replaced.

The imbalance of our created dataset is depicted in Figure 3.1. We used this text data augmentation to verify if it improves the model performance. Before data augmentation, we split the data into the train and validation set so that no samples in the validation set have been used for data augmentation. After performing this augmentation, we have achieved the class label counts as Figure 3.2.

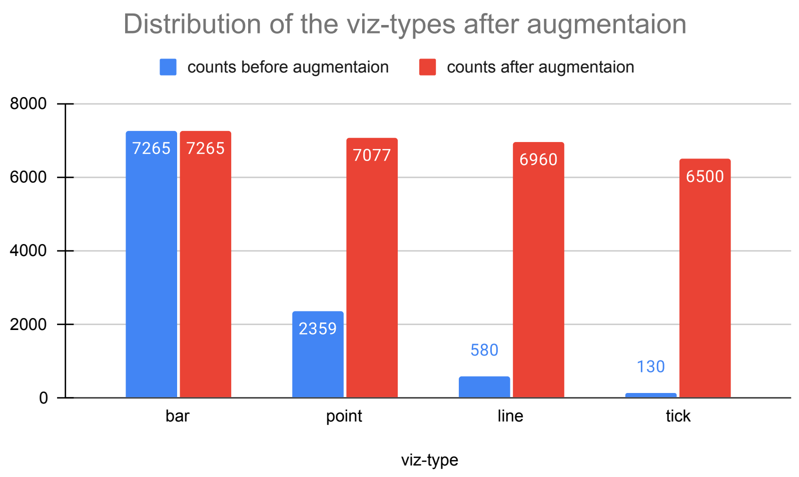


Figure 3.2: Distribution of augmented dataset

3.2.2 Text Preprocessing

Text preprocessing is a crucial step in the classification process. This step may help with classification precision. Preprocessing is used to eliminate noise and non-meaningful words from data. The following techniques were applied to each query item in the dataset:

1. Expanding the short forms in the sentences

2. Removing extra spaces and punctuations

3. Removing common stop-words
4. Lemmatization instead of stemming

3.3 Model Development for NLI to perform DV

3.3.1 Approach with rule-based model

The rule-based technique uses a pre-defined rule set to classify text into distinct categories, and thus requires a comprehensive domain expertise. Developing visualization NLIs is a difficult task which requires a low-level implementation of natural language processing (NLP) tools as well as a thorough understanding of visual analytic tasks and visualization design. For this project, we utilized NL4DV, a toolbox for natural language-driven data visualization. NL4DV is a Python tool that accepts a tabular dataset as well as a natural language query about it as input. The toolkit responds by returning an analytic specification in the form of a JSON object with data properties, analytic tasks, and a list of Vega-Lite specifications relevant to the input query.

During the parsing the query, this toolkit should identify the key information mentioned in the query explicitly or implicitly about the visualization. This extracted information can be visualized in the way depicted in Figure 3.3:

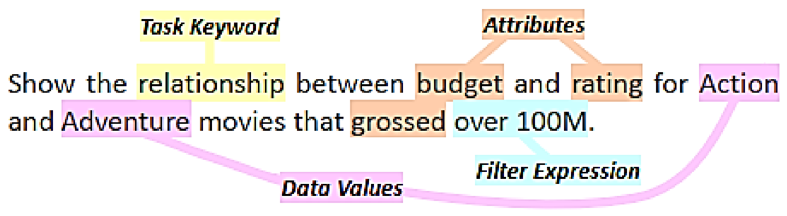
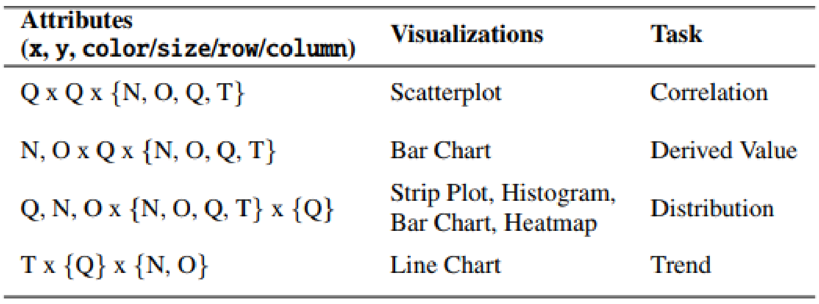

Task Keyword
Attributes
Show the relationship between budget and rating for Action
and Adventure movies that grossed over 100M.
Data Values
Filter Expression

Figure 3.3: Extracted information from the given NLQ [5]

After extracted the necessary information, this toolkit can analyze them with the help of a mapping table to draw the correct visualization. Here, NL4DV follows a mapping table like in Figure 3.4 for attribute, visualization and task mapping where Q, N, T, O refer to the data type elaborated as quant, nominal, order, and temporal.



Attributes (x, y, color/size/row/column)	Visualizations	Task
Q x Q x {N, O, Q, T}	Scatterplot	Correlation
N, O x Q x {N, O, Q, T}	Bar Chart	Derived Value
Q, N, O x {N, O, Q, T} x {Q}	Strip Plot, Histogram, Bar Chart, Heatmap	Distribution
T x {Q} x {N, O}	Line Chart	Trend

Figure 3.4: Mapping table of NL4DV [5]

3.4 Approach with ML based model

In this case we have to use NLP toolkit (e.g., NLTK) to parse all the key words into tokens. To implement this ML approach, we will go through the following steps:

3.4.1 Text Preprocessing

A typical pipeline for preprocessing text consists of the following steps:

1. **Sentence segmentation:** The text is divided into sentences in the first phase of the preprocessing procedure. Punctuation, particularly the full stop/period letter, exclamation, and question marks, may be used to indicate the conclusion of a sentence in many languages, including English. The period symbol, on the other hand, may be used in abbreviations like Ms. or U.K., in which case the full stop character does not indicate the conclusion of a sentence. To prevent misclassification of sentence borders, a table of abbreviations is utilized in certain circumstances. To prevent unnatural tokens, a separate dictionary of abbreviations must be created when the text contains domain-specific terminology.

2. **Normalization and tokenization:** Tokenization is the process of breaking down a text into words and punctuation marks, or tokens. Punctuation markers, like sentence segmentation, may be difficult to understand. For example, the United Kingdom should be treated as a single token, however ”don’t” should be divided into two tokens: ”do” and ”n’t.” The normalization procedure includes steps like as stemming and lemmatization. Stemming and lemmatization are two steps in normalization. The stem of the word is found

during the stemming process by eliminating suffixes like –ed and –ing. The resultant stem isn't always a word. Lemmatization is similar to deprecation in that it removes prefixes and suffixes, but the end product belongs to the language. A lemma is the name for this outcome.

3.4.2 Feature Extraction

The Bag-of-Words (BoW) technique is the simplest way to transform texts into numeric vectors. The basic idea behind BoW is to collect all of the unique words in a text corpus termed vocabulary. Each phrase may be represented as a vector of ones and zeros using the vocabulary, based on whether or not a word from the vocabulary is contained in the sentence.

3.4.3 Model development

We begin by separating the data into two sets: training and testing. Preprocessing and normalization of the train and test data is required before features can be retrieved. The most common text data feature extraction approaches were addressed in the preceding sections. Machine learning techniques may be used to text data after it has been transformed to numeric form. This is known as training the model, and it involves the model learning patterns from the characteristics in order to predict the labels. A procedure known as hyperparameter tuning may be used to improve the model's performance by tweaking model parameters.

There are several machine learning methods that are successful in the NLP arena. We've looked at Support Vector Machine (SVM), Naive Bayes (NB), Random Forest (RF), Gradient Boosting (GBT), Extreme Gradient Boosting (XGB) algorithms which separates classes by using different effective kernels. The goal of this kernel is to increase the distance between the data points and the hyperplane. Different regularization parameters are also added to the cost function. The regularization parameter's goal is to strike a compromise between accuracy and loss.

3.5 Approach with LSTM-RNN based model

3.5.1 Tokenizing the words and word embedding

The input sample messages are tokenized, transforming them into a sequence of tokens that the neural network uses as input. Tokens are also used to build a vocabulary of terms that will be utilized throughout training. We'll convert each query into an actual vector domain using word embedding using word2vec, a common text processing approach. This is a method in which words are stored as real-valued vectors in a high-dimensional space, with similarity in meaning translating to similarity in the vector space. Each word will be mapped to a 64-length real-valued vector.

3.5.2 Padding

As our queries are not of the same length, we padded zeroes to maintain a specific length. This sequence length is same as number of time steps for LSTM layer.

3.5.3 Train, test split

When we have got our dataset, we will split our dataset into training and test sets

3.5.4 Model development

We build our LSTM model into different layers following the steps mentioned below:

1. **Embedding layer:** It coverts our word tokens into embedding of specific size. To prepare this embedding, it will create a neural network to update the weights based on the corpus of V-NLIs we are providing. It takes two parameters: the vocabulary size and the dimensionality of the embedding. From the output of embedding layer, we can see it has created a 3-dimensional tensor as a result of embedding weights. Now it has train-samples number of rows, 100 columns and 30 embedding dimensions i.e. for each tokenized word in the query we have added embedding dimension. This data will now go to LSTM Layer.

2. **LSTM Layer:** It defines the hidden state dimensions and number of layers. In this case, we are using a single LSTM layer of 100 neurons (also known as hidden units). This layer is properly maintained with appropriate batch size of inputs and drop out threshold value.

3. **Fully connected Layer:** For fully connected layer, number of input features is equal to the number of hidden units in LSTM. It maps the outputs of LSTM layer into the desired output size based on the number of class labels we are providing. Here, this model will predict 4 different viz-types. Note that before putting the LSTM output into the fully connected layer, it has to be flattened out. Next, we use softmax activation function that turns all output values into the probabilities of four classes.

4. **Output:** Softmax function from the last timestamp is considered as the final output of the network. The outputs from this network are from an untrained network for the first epoch. Hence, the output values will be more and more accurate based on the number epochs we will train this network and it will try to minimize its loss based on an optimizer approach.

3.6 Model Development for XAI: LIME Pipeline

In this pipeline we are going to explain our developed LSTM model in last section in depth. To build this pipeline we do the following steps as we have mentioned in the section 2.3.1:

1. Derive the permutation of each test case to be explained
2. Utilize the given complex model to predict all permuted test cases
3. Calculate the distance (similarity) between the permutated and original test cases and convert them into similarity scores
4. Make the sub-set of K features with highest importance in complex model for each permuted test case around the given instance
5. Fit a linear model with the sub-set of K features based on the permuted data and derive the weights for this model.
6. Using this linear model explain the given instance prediction

Mathematically, the local surrogate models with interpretability constraint can be expressed as follows which is an array of weights of the linear model:

$$explanation(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g) \tag{3.1}$$

The explanation model for instance x is the model g (e.g. linear regression model) that minimizes loss L (e.g. mean squared error), which measures how close the explanation is to the prediction of the original model f (e.g. an xgboost model), while the model complexity $\Omega(g)$ is kept low (e.g. prefer fewer features). G is the family of possible explanations, for example all possible linear regression models. The proximity measure π_x defines how large the neighborhood around instance x is that we consider for the explanation. In practice, LIME only optimizes the loss part. The user has to determine the complexity, e.g. by selecting the maximum number of features that the linear regression model may use.

To apply this XAI pipeline we will do the following steps:

1. **Surrogate Linear Model Development for Explanation:**

   Implement the explanation function using the following pseudocode:

Algorithm 2 Sparse Linear Explanations using LIME

```
1: Require: Classifier $f$, Number of samples $N$
2: Require: Instance $x$, and its interpretable version $x'$
3: Require: Similarity kernel $\pi_x$, Number of features in the explanation $K$
4: $Z \leftarrow \{\}$
5: for $i \in \{1, 2, 3, \dots, N\}$ do
6:     $z'_i \leftarrow sample\_around(x')$
7:     $Z \leftarrow Z \cup ((z'_i, f(z_i), \pi_x(z_i)))$
8: end for
9: $w \leftarrow fidelity\_function(Z, K)$    // weight update minimizing the loss
10: return $w$    // updated weight
```

   Here, G is a class of interpretable models such as linear models, decision tree. So $g \in G$ can be readily presented to the user visually or textually. Here, the returned values w represents the linear model g. Let f be the model to be explained and x is an instance and $f(x)$ is a probability to a certain class. We use $\pi_x(z)$ as a proximity measure between x and z to define the locality of x. L is the loss function which should be minimized having the complexity

Ω (opposed to interpretability) of g. Finally, let $L(f, g, \pi_x)$ be a measure of how unfaithful g is in approximating f in the locality defined by π_x.

2. **Explaining the prediction on a particular instance:**

Here, we will provide the explanation of an instance based on the graphical interface. First, an explanation instance will be created based on the explanation object created in the last step to perturbate the samples based on the given instance. With this instance, LIME will produce an interpretable linear model to explain a particular instance selecting several top features and several neighborhood samples. This interpretable linear model could be used to present the visualization of the explanation.

Chapter 4

Experimental Design

To develop our system, we carefully evaluate each stages described as proposed methodology in Chapter 3 and explain each alternative choice of our implementation in Chapter 5 for justification. For the implementation of the whole project we have divided our task into two parts where in the first we have implemented the deep learning based NLP model to extract key information from the given query and in the later half, we have implemented the pipeline to trnsform a black-box AI model into white-box.

4.1 LSTM Implementation

- **Environment:** The proposed system was implemented in Google Colab using Tensorflow-Keras framework (version 2.7.1).The environment provides an NVIDIA Tesla T4 GPU with a VRAM of 16 GB and an Intel Xeon CPU with a base clock speed of 2.3 GHz. The total usable memory of the system was 12 GB.
- **Dataset Split:** We have used the preprocessed QUDA dataset which contains 10334 queries with 4 types of visualization types (i.e., bar, point, line and tick). From that initial dataset, we have split it in 80:20 ratio as train and test set. We have used this test set to evaluate the accuracy of all the models. In the end, we have 8266 train queries and 2068 test queries. During the training we have split the dataset further in 80:20 ratio for training and development purpose where we have 6613 and 1653 queries respectively following the approach of the experiment [73].

- **Dataset augmentation:** The insufficient data in different class labels is one of the most common challenges in accurately implementing such models. We have implemented the synonym replacement based data augmentation approach via word embeddings [71]. We replace n number words with its synonyms (word embeddings that are close to those words) to obtain a sentence with the same meaning but with different words. This n depends on the number of iterations we have made to paraphrase each query based on a particular class label.
- **Hyper-parameter Tuning** To reduce the overfitting problem of the model we have explored different drop-out values between 0.2 and 0.5. The batch size was 32 and number of hidden states in the LSTM layer were 100. The vocabulary size was set 5000 and the maximum query length was set 50 analyzing all the queries in the dataset.
- **Optimizer Tuning:** We have used ADAM optimizer with hyper-parameters β_1 set to 0.9 and β_2 set to 0.999 for optimization. The initial learning rate is set to be 0.1 with the decay ratio of 0.95. If the validation loss starts to be increased it reduces its learning rate and before overfitting the model it stops its training as an early stopping approach following the method of the experiment [73].
- **Methods for Comparison:** We have compared our model with NL4DV as a rule based model, Additionally, we compared our model with Support Vector Machine (SVM), Naive Bayes (NB), Random Forest (RF), Gradient Boosting (GBT), Extreme Gradient Boosting (XGB) as the ML models since these models are frequently used in NLP related experiments [27]. We re-implemented those models and reported the performance of our implementation.

4.2 LIME Implementation

To setup LIME we import the relevant libraries and make sure that our data and model are in the right format to input into the LIME explainer. This means that our data should be a list or array of strings and we need to create a pipeline which takes in our data, and both transforms the text data with the vectoriser we already built and then makes the predictions with the trained LSTM model. Next, we create our explainer and choose our single prediction which we would like to be explained. The feature weights are calculated using the LIME explanation. In the *explain_instance* algorithm we could also add the parameter *num_features* if we wanted to reduce the number of features used to explain the prediction.

Chapter 5

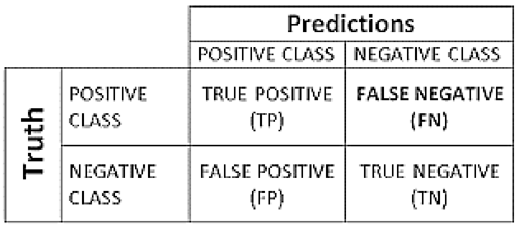
Results and Discussions

5.1 Evaluation Metrics

Evaluation metrics play a crucial role in quantifying the performance of a predictive classifier. Since the choice of metrics depends on the characteristic of the dataset, this can often lead to misleading conclusion regarding the experiment. For example, while evaluating an experiment on a highly imbalanced dataset, evaluation metrics such as accuracy, precision, or recall may lead to a conclusion that is practically useless. With imbalanced datasets, it is possible to reach very high accuracy without predicting any useful prediction since the majority predictions are from the densely populated classes.

Other widely used evaluation metrics like precision, recall, F1 score, ROC-AUC etc based on confusion matrix shown in Figure 5.1. have their own limitations. Precision is about exactness of classification task and relies only on true positive and false positive, it is possible to get a precision score of 1.0 by only one true positive prediction. On the other hand, recall is about completeness and depends solely on true positive and false negative. As a result, predicting all the samples as positive will give a recall of 1.0, whereas precision will be very low.

To tackle this issue, the Receiver Operating Characteristic (ROC) curve and area under the ROC curve (AUC-ROC) are used as evaluation measures in this work, such that models are evaluated based on how good they are at separating classes. ROC curve is a diagnostic diagram that calculates the False Positive Rate (FPR), and True Positive Rate (TPR) for a series of predictions made by the model at different thresholds to summarize the model's behavior which can be used to



		Predictions	
		POSITIVE CLASS	NEGATIVE CLASS
Truth	POSITIVE CLASS	TRUE POSITIVE (TP)	**FALSE NEGATIVE (FN)**
	NEGATIVE CLASS	FALSE POSITIVE (FP)	TRUE NEGATIVE (TN)

Figure 5.1: A basic confusion matrix

analyze the model's ability to discriminate classes. True Positive Rate (TPR) tells what proportion of the positive class get correctly classified by the classifier. False Positive Rate (FPR) tells what proportion of the negative class got incorrectly classified by the classifier. TPR and FPR are calculated as follows:

$$TPR = \frac{TP}{ActualPositive} = \frac{TP}{TP + FN} \tag{5.1}$$

$$FNR = \frac{FN}{ActualPositive} = \frac{FN}{TP + FN} \tag{5.2}$$

$$TNR = \frac{TN}{ActualNegative} = \frac{TN}{TN + FP} \tag{5.3}$$

$$FPR = \frac{FP}{ActualNegative} = \frac{FP}{TN + FP} \tag{5.4}$$

The Receiver Operator Characteristic (ROC) curve is a probability curve that plots the TPR against FPR at various threshold values and essentially separates the 'signal' from the 'noise'. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. A model that with no discriminatory power between the classes will be represented by a diagonal line between FPR 0 and TPR 0 (co-ordinate: 0,0) to FPR 1 and TPR 1 (co-ordinate: 1,1). Points below this line reflect models with less competence than none. A flawless model will be represented as a point in the plot's upper left corner.

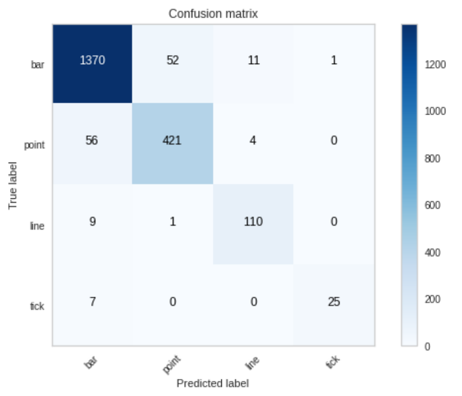
The final accuracy of the models we have tried on the test data set are depicted in the table 5.1:

Table 5.1: Accuracy per model

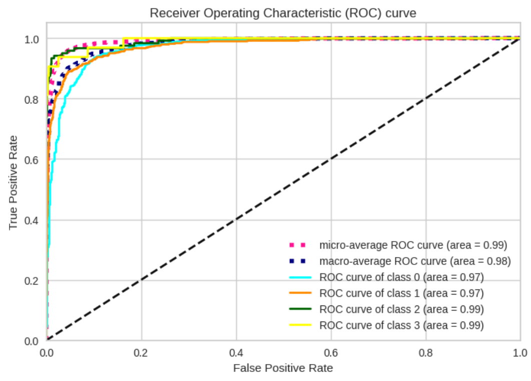
	Bar-F1 Score	**Point-F1 Score**	**Line-F1 Score**	**Tick-F1 Score**	**Test Accura**
NL4DV	-	-	-	-	0.503
GBT	0.935	0.858	0.828	0.875	**0.909**
NB	0.848	0.730	0.792	0.690	0.806
RF	0.019	0.402	0.705	0.779	0.292
SVM	0.930	0.846	0.890	0.852	**0.905**
XGB	0.921	0.810	0.806	0.825	0.887
LSTM-V	0.944	0.868	0.884	0.800	**0.922**
LSTM-P	0.947	0.869	0.885	0.800	**0.924**

5.2 Result Analysis of LSTM

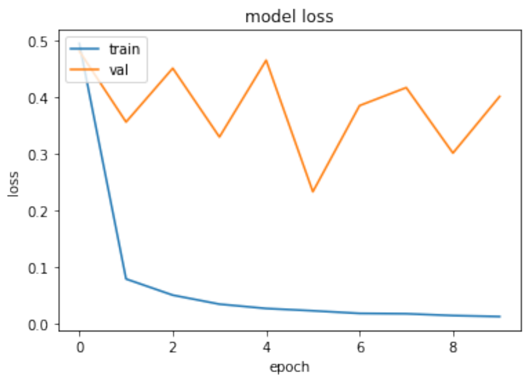
Here are the evaluation graphs considering no drop-outs during the training of the LSTM model in Figure 5.3:



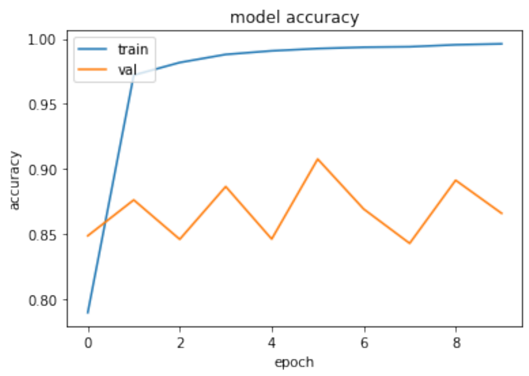
(a) Confusion matrix



(b) ROC-AUC Curve



(c) Loss graph

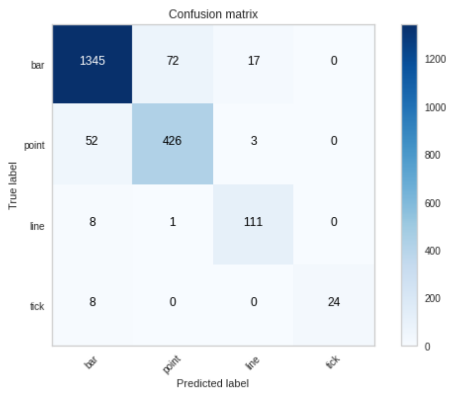


(d) Accuracy graph

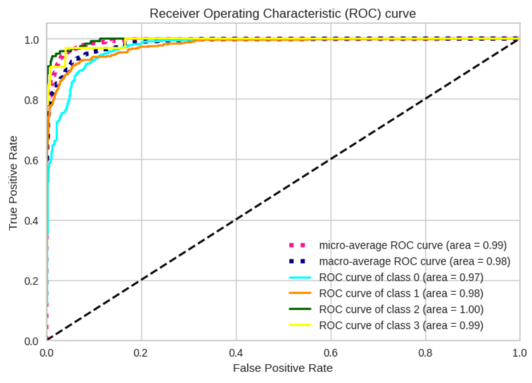
Figure 5.2: Evaluation graphs - without dropouts

In Figure 5.3a, we can see the confusion matrix considering no dropouts. As it over-fits the model, we can the loss graph in Figure 5.3c and the accuracy graph in 5.3d is not satisfactory.

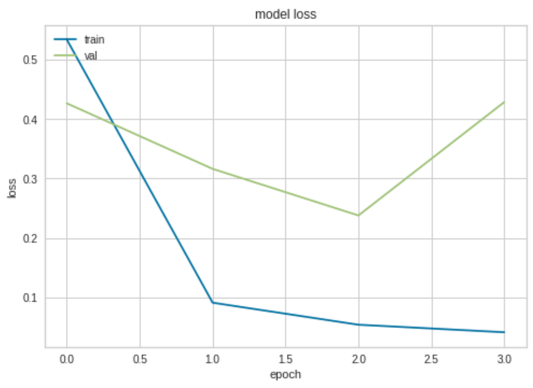
Then we have added the drop outs (value=0.2) after each LSTM and dense layer to reduce the bias of the model and achieved the following results in Figure 5.3:.



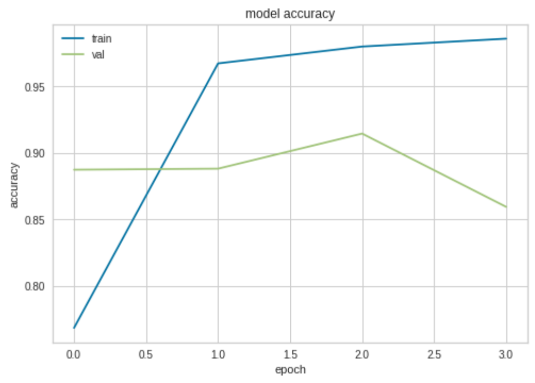
(a) Confusion matrix



(b) ROC-AUC Curve



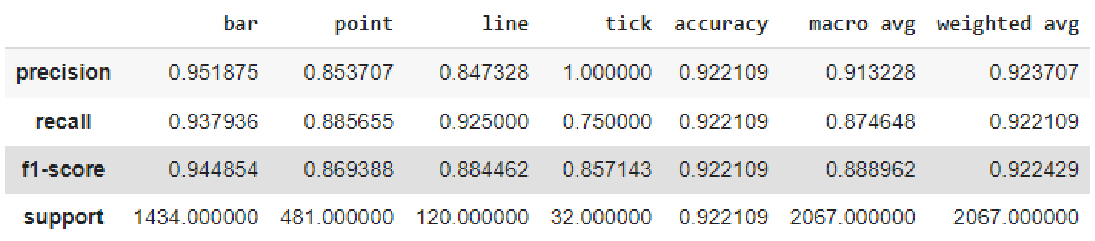
(c) Loss graph



(d) Accuracy graph

Figure 5.3: Evaluation graphs - with dropouts

Here, we can see the randomness of the loss and accuracy graph is removed as the overfitting problem of the LSTM model is removed. Also this has improved the F1 scores which is reflected in the confusion matrix, The overall classification report is depicted in Figure 5.4.



	bar	point	line	tick	accuracy	macro avg	weighted avg
precision	0.951875	0.853707	0.847328	1.000000	0.922109	0.913228	0.923707
recall	0.937936	0.885655	0.925000	0.750000	0.922109	0.874648	0.922109
f1-score	0.944854	0.869388	0.884462	0.857143	0.922109	0.888962	0.922429
support	1434.000000	481.000000	120.000000	32.000000	0.922109	2067.000000	2067.000000

Figure 5.4: Classification report - final

5.3 Explanation of the produced model using LIME

The LIME pipeline helps to identify an interpretable model over the interpretable representation locally. When we apply LIME for an explanation of individual predictions, it shows the visualization type output results with each featuring tokenized word from a given query. In numerical contribution, the LSTM model gives a particular contributing scores to each token based on probability during the training of the model.

LIME is also capable of local interpretability of the models. Figures 5.5-5.8 show the local explanations for different user given NL queries predicting different visualization types.

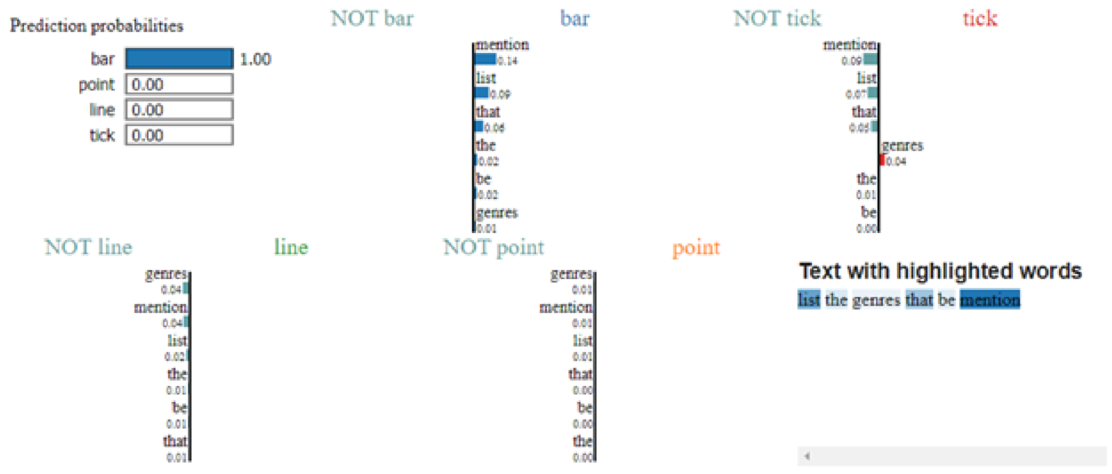


Figure 5.5: LIME explanation for "**bar**" visualization type

Figure 5.5 shows the contributing tokens to visualize a chart as a bar chart. Here, it finds "*list*" word contributing for such graph type. From the common knowledge, it is obvious that it is true most of the time in real world and the model learns in this way during its training.

Figure 5.6 shows "*years*" token contributing to visualize a line chart. In the real world, it is very common that we depict any time related trend with line chart. Such this way, our model learned this pattern during training.

Figure 5.7 shows "*birdcall*" token contributing to visualize a line chart. In the real world, it is very common that we depict any one dimension feature in different range using strip plot. So here each birdcall is made of different duration. Such this way, our model learned this pattern during training.

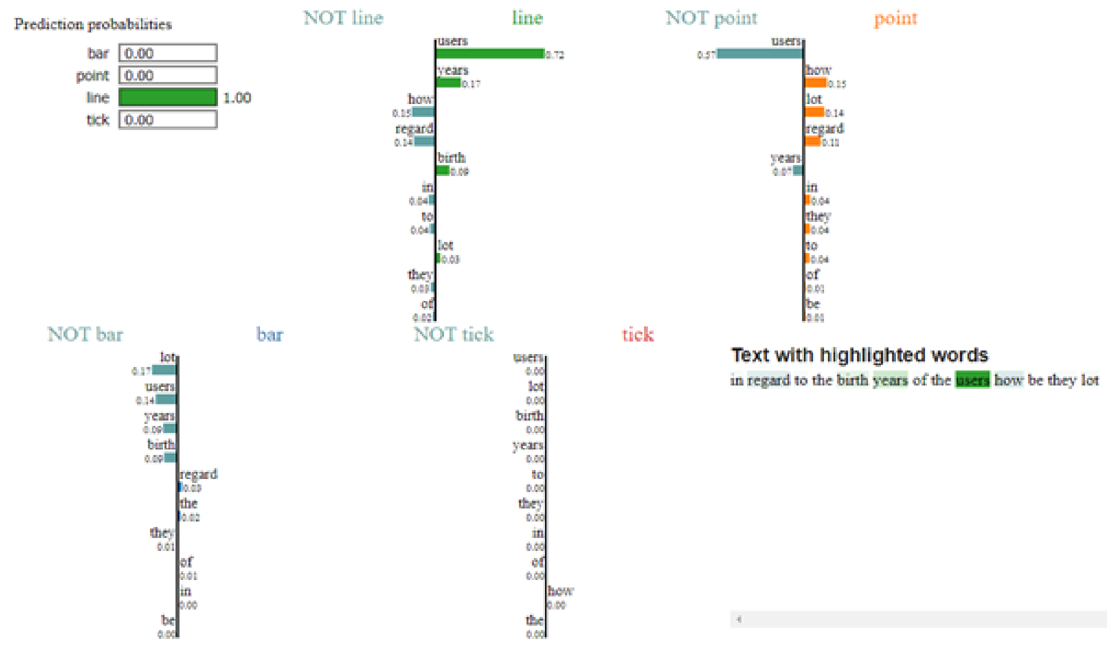


Figure 5.6: LIME explanation for "**line**" visualization type

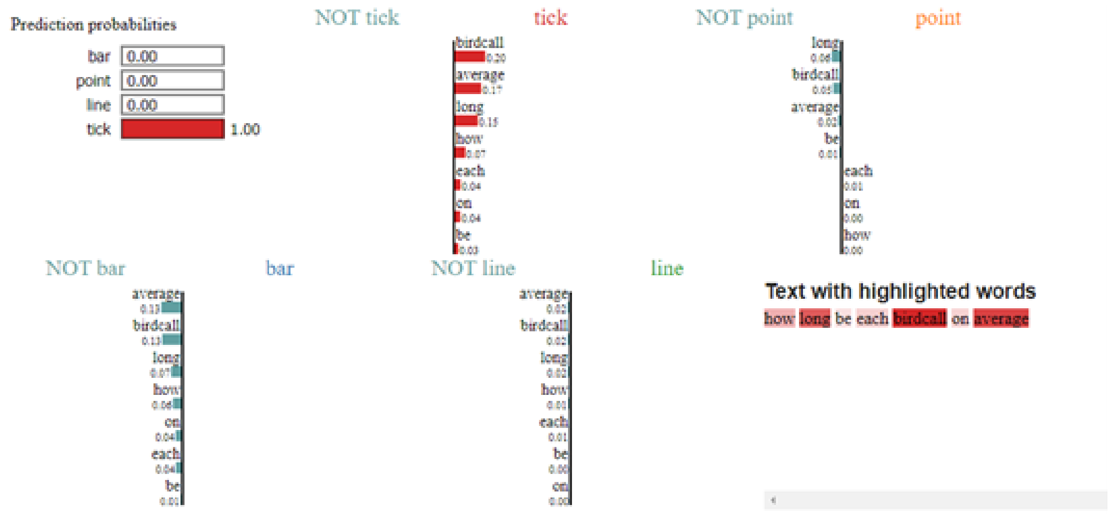


Figure 5.7: LIME explanation for "**tick**" visualization type

In Figure 5.8 finds "*grade*" word contributing for line graph type. In general, we show the randomness of grades with scatter plots in data visualization. This model learns this pattern to visualize data in this way during training period.

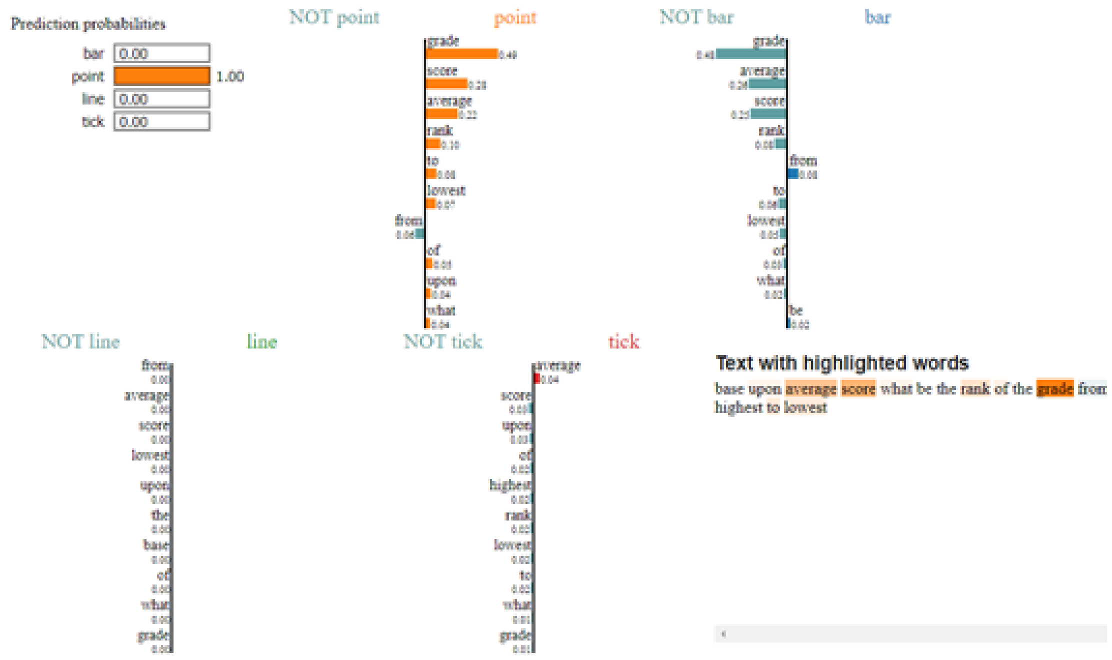


Figure 5.8: LIME explanation for "**point**" visualization type

5.4 Discussion

5.4.1 LSTM model on NLP process

The rule-based approach did really the bad prediction when we have paraphrased the queries. It is because it only checks the presence of certain words in the query to decide and if the query structure doesn't match with the mapping function it mispredicts that query though it has same connotation with seed query where NL4DV works well. It is little bit improved when we have used ML approach, but it is not satisfactory because still it checks the presence of words and word to word sematic sequence is not maintained. But LSTM does maintain the word-to-word semantic sequence and performs better for the paraphrases.

But in computation power analysis, rule-based NL4DV approach does not give any load to learn any parameters whereas SVM will maintain the kernel function to make the decision. In the LSTM model,it has around 213636 learnable parameters which causes a large amount of computation power in exchange of best accuracy.

5.4.2 LIME: explain an ML model

In some cases, our LSTM model predicts the instance correctly, but for the wrong reasons. In train set, the word “List” comes in most of the cases of “bar chart” comparing other visualization types and happened the most similarly for test dataset as well. This kind of artifact in the data set makes the problem much easier than it is in the real world, where we wouldn’t expect such patterns to occur. Here we can observe that both machine learning experts and novice users greatly could benefit from the explanations like the given examples. Moreover, it could enable them to choose which models generalize better, improve models by changing them, and get crucial insights into the models’ behavior.

The correct definition of the neighborhood is a very big, unsolved problem when using LIME with tabular data. In my opinion it is the biggest problem with LIME and the reason why I would recommend using LIME only with great care. For each application you must try different kernel settings and see for yourself if the explanations make sense. Unfortunately, this is the best advice I can give to find good kernel widths.

Sampling could be improved in the current implementation of LIME. Data points are sampled from a Gaussian distribution, ignoring the correlation between features. This can lead to unlikely data points which can then be used to learn local explanation models.

Chapter 6

Conclusion and Future Work

In this research work, first, we propose a pipeline to develop a V-NLI based deep learning model and extract key information for appropriate data visualization. Next, we have extended that pipeline to make the decision for data visualization for the user explainable in order to earn the user's trust and make them transparent. According to our proposed method, we attempt to boost the efficiency of our model by utilizing the LSTM language model to extract relevant information from user input¿ in this regard we did not rely on the presence of certain words to make the decision rather we have checked the semantic notation exploring the relations among the words in a sentence sequence.

Trust is an another crucial for effective human interaction with machine learning systems, and we think explaining individual predictions is an effective way of assessing trust. We have used the LIME technique to produce a visual explanation of the system's decision. Our proposed pipeline is an efficient tool to facilitate such trust for machine learning practitioners and a good choice to add to their tool belts, but there is still plenty of work to be done to better explain machine learning models.

References

[1] S. Fu, K. Xiong, X. Ge, S. Tang, W. Chen, and Y. Wu, "Quda: Natural Language Queries for Visual Data Analytics," *CoRR*, vol. abs/2005.03257, 2020. [Online]. Available: https://arxiv.org/abs/2005.03257

[2] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer, "Vega-Lite: A Grammar of Interactive Graphics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 341–350, 2017. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7539624

[3] C. Liu, Y. Han, R. Jiang, and X. Yuan, "ADVISor: Automatic Visualization Answer for Natural-Language Question on Tabular Data," in *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*. Tianjin, China: IEEE, 2021, pp. 11–20. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9438784

[4] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[5] A. Narechania, A. Srinivasan, and J. Stasko, "NL4DV: A Toolkit for Generating Analytic Specifications for Data Visualization from Natural Language Queries," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 369–379, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9222342

[6] I. Chowdhury, A. Moeid, E. Hoque, M. A. Kabir, M. S. Hossain, and M. M. Islam, "MIVA: Multimodal Interactions for Facilitating Visual Analysis with Multiple Coordinated Views," in *2020 24th International Conference Information Visualisation (IV)*. Melbourne, Australia: IEEE, 2020, pp. 714–717. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9373232

[7] M. T. Islam, M. R. Islam, S. Akter, and M. Kawser, "Designing Dashboard for Exploring Tourist Hotspots in Bangladesh," in *2020 23rd International Conference on Computer and Information Technology (ICCIT)*. Dhaka, Bangladesh: IEEE, 2020, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9392708

[8] M. R. Islam, S. Akter, M. R. Ratan, A. R. M. Kamal, and G. Xu, "Deep Visual Analytics (DVA): Applications, Challenges and Future Directions," *Human-Centric Intelligent Systems*, vol. 1, pp. 3–17, 2021. [Online]. Available: https://www.atlantis-press.com/journals/hcis/125959056/view

[9] J. Zerafa, M. R. Islam, A. Kabir, and G. Xu, "ExTraVis: Exploration of Traffic Incidents Using Visual Interactive System," in *25th International Conference Information Visualisation (IV 2021)*, IEEE, Institute of Electrical and Electronics Engineers. Sydney, Australia: IEEE, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9582720

[10] B. Lee, P. Isenberg, N. H. Riche, and S. Carpendale, "Beyond mouse and keyboard: Expanding design considerations for information visualization interactions," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2689–2698, dec 2012. [Online]. Available: https://doi.org/10.1109%2Ftvcg.2012.204

[11] R. Amar, J. Eagan, and J. Stasko, "Low-level components of analytic activity in information visualization," in *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.* Minneapolis, MN, USA: IEEE, 2005, pp. 111–117. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1532136

[12] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer, "Reactive Vega: A Streaming Dataflow Architecture for Declarative Interactive Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 659–668, 2016. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7192704

[13] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," 2017, to appear.

[14] V. Setlur, M. Tory, and A. Djalali, "Inferencing Underspecified Natural Language Utterances in Visual Analysis," in *Proceedings of the 24th*

International Conference on Intelligent User Interfaces, ser. IUI '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 40–51. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3301275.3302270

[15] D. H. Kim, E. Hoque, and M. Agrawala, "Answering Questions about Charts and Generating Visual Explanations," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–13. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3313831.3376467

[16] A. Kumar, S. Dikshit, and V. H. C. Albuquerque, "Explainable Artificial Intelligence for Sarcasm Detection in Dialogues," *Wireless Communications and Mobile Computing*, vol. 2021, 2021. [Online]. Available: https://www.hindawi.com/journals/wcmc/2021/2939334/

[17] M. R. Islam, S. Liu, X. Wang, and G. Xu, "Deep learning for misinformation detection on online social networks: a survey and new perspectives," *Social Network Analysis and Mining*, vol. 10, no. 1, pp. 1–20, 2020. [Online]. Available: https://link.springer.com/article/10.1007/s13278-020-00696-x

[18] K. Cox, R. E. Grinter, S. L. Hibino, L. J. Jagadeesan, and D. Mantilla, "A Multi-Modal Natural Language Interface to an Information Visualization Environment," *International Journal of Speech Technology*, vol. 4, no. 3, pp. 297–314, 2001. [Online]. Available: https://link.springer.com/article/10.1023/A:1011368926479

[19] 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 28th Annual ACM Symposium on User Interface Software & Technology*, ser. UIST '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 489–500. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2807442.2807478

[20] 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*, R. Taylor, P. Boulanger, A. Krüger, and P. Olivier, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 184–195. [Online]. Available: https://rd.springer.com/chapter/10.1007/978-3-642-13544-6_18

[21] B. Yu and C. T. Silva, "FlowSense: A Natural Language Interface for Visual Data Exploration within a Dataflow System," *IEEE Transactions*

on Visualization and Computer Graphics, vol. 26, no. 1, pp. 1–11, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8807265

[22] V. Setlur, S. E. Battersby, M. Tory, R. Gossweiler, and A. X. Chang, "Eviza: A natural language interface for visual analysis," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, ser. UIST '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 365–377. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2984511.2984588

[23] E. Hoque, V. Setlur, M. Tory, and I. Dykeman, "Applying Pragmatics Principles for Interaction with Visual Analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 309–318, 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8019833

[24] L. Shen, E. Shen, Y. Luo, X. Yang, X. Hu, X. Zhang, Z. Tai, and J. Wang, "Towards natural language interfaces for data visualization: A survey," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2022. [Online]. Available: https://doi.org/10.1109%2Ftvcg.2022.3148007

[25] J. Hu, "12explainable deep learning for natural language processing," Ph.D. dissertation, KTH Royal Institute of Technology, 2018. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-254886

[26] T. Spinner, U. Schlegel, H. Schäfer, and M. El-Assady, "explAIner: A Visual Analytics Framework for Interactive and Explainable Machine Learning," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 1064–1074, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8807299

[27] M. Danilevsky, K. Qian, R. Aharonov, Y. Katsis, B. Kawas, and P. Sen, "A Survey of the State of Explainable AI for Natural Language Processing," in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*. Suzhou, China: Association for Computational Linguistics, Dec. 2020, pp. 447–459. [Online]. Available: https://aclanthology.org/2020.aacl-main.46

[28] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,*

Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[29] A. Kumar Sharma, S. Hazra, and A. Professor, “10. Application of Deep Learning Techniques for Text Classification on Small Datasets,” *International Journal of Engineering Science and Computing*, vol. 8, no. 4, pp. 17 212–17 213, 2018. [Online]. Available: http://ijesc.org/

[30] H. Liu, Q. Yin, and W. Y. Wang, “Towards Explainable NLP: A Generative Explanation Framework for Text Classification,” Tech. Rep. [Online]. Available: https://www.pcmag.com/

[31] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” *Advances in neural information processing systems*, vol. 28, 2015.

[32] M. Danilevsky, K. Qian, R. Aharonov, Y. Katsis, and P. Sen, “A Survey of the State of Explainable AI for Natural Language Processing,” Tech. Rep. [Online]. Available: https://xainlp2020.github.io/xainlp/

[33] M. A. Yalcin, N. Elmqvist, and B. B. Bederson, “Keshif: Rapid and expressive tabular data exploration for novices,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 8, pp. 2339–2352, aug 2018. [Online]. Available: https://doi.org/10.1109%2Ftvcg.2017.2723393

[34] M. Vartak, S. Madden, A. Parameswaran, and N. Polyzotis, “SeeDB,” *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1581–1584, aug 2014. [Online]. Available: https://doi.org/10.14778%2F2733004.2733035

[35] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer, “Voyager: Exploratory analysis via faceted browsing of visualization recommendations,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 649–658, jan 2016. [Online]. Available: https://doi.org/10.1109%2Ftvcg.2015.2467191

[36] P. L. M. Olivia Nix, “Now in beta: Ask questions of your data with natural language, schedule your tableau prep flows,” Oct 2018. [Online]. Available: https://www.tableau.com/

[37] “Data visualization: Microsoft power bi.” [Online]. Available: https://powerbi.microsoft.com/en-us/

[38] F. B. Viegas, M. Wattenberg, F. van Ham, J. Kriss, and M. McKeon, “ManyEyes: a site for visualization at internet scale,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1121–1128, nov 2007. [Online]. Available: https://doi.org/10.1109%2Ftvcg.2007.70577

[39] C. Stolte and P. Hanrahan, “Polaris: a system for query, analysis and visualization of multi-dimensional relational databases,” in *IEEE Symposium on Information Visualization 2000. INFOVIS 2000. Proceedings.* IEEE Comput. Soc. [Online]. Available: https://doi.org/10.1109%2Finfvis.2000.885086

[40] A. Satyanarayan and J. Heer, “Lyra: An interactive visualization design environment,” *Computer Graphics Forum*, vol. 33, no. 3, pp. 351–360, jun 2014. [Online]. Available: https://doi.org/10.1111%2Fcgf.12391

[41] D. Ren, T. Hollerer, and X. Yuan, “iVisDesigner: Expressive interactive design of information visualizations,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2092–2101, dec 2014. [Online]. Available: https://doi.org/10.1109%2Ftvcg.2014.2346291

[42] M. Bostock, V. Ogievetsky, and J. Heer, “D^3 Data-Driven Documents,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2301–2309, 2011. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6064996

[43] L. Wilkinson, *The Grammar of Graphics.* Springer New York, 1999. [Online]. Available: https://doi.org/10.1007%2F978-1-4757-3100-2

[44] H. Wickham, “ggplot2,” *Wiley interdisciplinary reviews: computational statistics*, vol. 3, no. 2, pp. 180–185, 2011.

[45] W. Chang and H. Wickham, “ggvis: Interactive grammar of graphics,” *R package version0*, vol. 4, 2016.

[46] K. Cox, R. E. Grinter, S. L. Hibino, L. J. Jagadeesan, and D. Mantilla, “A multi-modal natural language interface to an information visualization environment,” *International Journal of Speech Technology*, vol. 4, no. 3, pp. 297–314, 2001.

[47] T. Ball, C. Colby, P. Danielsen, L. J. Jagadeesan, R. Jagadeesan, K. Läufer, P. Mataga, and K. Rehor, “Sisl: Several interfaces, single logic,” *International Journal of Speech Technology*, vol. 3, no. 2, pp. 93–108, 2000.

[48] K. Dhamdhere, K. S. McCurley, R. Nahmias, M. Sundararajan, and Q. Yan, "Analyza: Exploring data with conversation," in *Proceedings of the 22nd International Conference on Intelligent User Interfaces*. ACM, mar 2017, pp. 493–504. [Online]. Available: https://doi.org/10.1145%2F3025171.3025227

[49] "Ibm analytics." [Online]. Available: https://www.ibm.com/au-en/analytics

[50] "Wolfram: Alpha." [Online]. Available: https://www.wolframalpha.com/

[51] "Thoughtspot; ai-driven analytics," Apr 2022. [Online]. Available: https://www.thoughtspot.com/

[52] E. Hoque, V. Setlur, M. Tory, and I. Dykeman, "Applying pragmatics principles for interaction with visual analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 309–318, 2018.

[53] A. Srinivasan and J. Stasko, "Orko: Facilitating Multimodal Interaction for Visual Exploration and Analysis of Networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 511–521, 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8019860

[54] Y.-H. Kim, B. Lee, A. Srinivasan, and E. K. Choe, "Data@ hand: Fostering visual exploration of personal data on smartphones leveraging speech and touch interaction," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–17.

[55] A. Srinivasan, B. Lee, N. H. Riche, S. M. Drucker, and K. Hinckley, "InChorus: Designing consistent multimodal interactions for data visualization on tablet devices," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, apr 2020. [Online]. Available: https://doi.org/10.1145%2F3313831.3376782

[56] W. Cui, X. Zhang, Y. Wang, H. Huang, B. Chen, L. Fang, H. Zhang, J.-G. Lou, and D. Zhang, "Text-to-viz: Automatic generation of infographics from proportion-related natural language statements," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 906–916, jan 2020. [Online]. Available: https://doi.org/10.1109%2Ftvcg.2019.2934785

[57] N. Ranjan, K. Mundada, K. Phaltane, and S. Ahmad, "A survey on techniques in NLP," *International Journal of Computer Applications*, vol. 134, no. 8, pp. 6–9, jan 2016. [Online]. Available: https://doi.org/10.5120%2Fijca2016907355

[58] Y. Kang, Z. Cai, C.-W. Tan, Q. Huang, and H. Liu, "Natural language processing (NLP) in management research: A literature review," *Journal of Management Analytics*, vol. 7, no. 2, pp. 139–172, apr 2020. [Online]. Available: https://doi.org/10.1080%2F23270012.2020.1756939

[59] 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.* Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 63–70. [Online]. Available: https://aclanthology.org/W02-0109

[60] "Corenlp overview." [Online]. Available: https://stanfordnlp.github.io/CoreNLP/

[61] "Apache opennlp overview." [Online]. Available: https://opennlp.apache.org/

[62] "Allennlp - allen institute for ai." [Online]. Available: https://allenai.org/allennlp

[63] "Gensim: Topic modelling for humans," May 2022. [Online]. Available: https://radimrehurek.com/gensim/

[64] "Textblob: Simplified text processing." [Online]. Available: https://textblob.readthedocs.io/en/dev/

[65] "Nlp architect: Simplified text processing." [Online]. Available: https://intellabs.github.io/nlp-architect/#

[66] "Googlenlp." [Online]. Available: https://github.com/BrianWeinstein/googlenlp

[67] flairNLP, "Flairnlp/flair: A very simple framework for state-of-the-art natural language processing (nlp)." [Online]. Available: https://github.com/flairNLP/flair

[68] "Stanza: Overview." [Online]. Available: https://stanfordnlp.github.io/stanza/

[69] B. Yu and C. T. Silva, "Visflow-web-based visualization framework for tabular data with a subset flow model," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 1, pp. 251–260, 2016.

[70] L. Shen, E. Shen, Y. Luo, X. Yang, X. Hu, X. Zhang, Z. Tai, and J. Wang, "Towards Natural Language Interfaces for Data Visualization: A Survey,"

IEEE Transactions on Visualization and Computer Graphics, vol. XX, no. X, pp. 1–20, 2022.

[71] J. Wei and K. Zou, “EDA: Easy data augmentation techniques for boosting performance on text classification tasks,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019. [Online]. Available: https://doi.org/10.18653%2Fv1%2Fd19-1670

[72] J. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi, “TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2020. [Online]. Available: https://doi.org/10.18653%2Fv1%2F2020.emnlp-demos.16

[73] S. Wang and J. Jiang, “Learning natural language inference with lstm,” *arXiv preprint arXiv:1512.08849*, 2015.