Data Visualization By Voice: Syntax, Parsing, and Recognition Techniques

Maurice Diesendruck
Department of Statistics and Data Science
momod@utexas.edu

Honghe Zhao
Department of Mathematics
joehonghe@utexas.edu

Code available at:
https://github.com/diesendruck/ggspeak

ABSTRACT

This research identifies techniques that enable a computer system to perform automated data visualization by actively "listening" to a user's natural spoken language, in an interactive and real-time session. This work coins the name *ggspeak* as the "grammar of graphics" for speech, and presents software that incorporates speech recognition and a domain-specific entity extraction algorithm that respects and resolves errors of mistranscription.

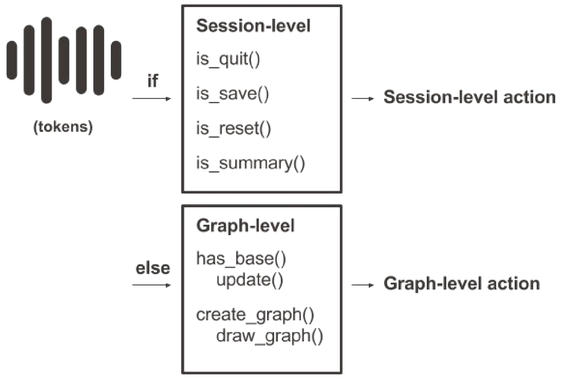
SPOKEN WORKFLOW

The syntax and pacing of speech differs greatly from that of the written language (coded syntax being one example). When speaking, people pause, repeat words, allude to things implicitly, and carry context from one statement to the next. Any voice system used to produce graphics would ideally respect these natural qualities of speech, and eventually produce a correct, detailed, and explicit definition for a graph object.

Ideal Conversation

"Show me the relationship between carat and price"
"Group by clarity"
"Save it"
"New graph"
"Scatter plot x and y"
"Color by carat"
"Save it"
"Quit"

Two-Level Hierarchy of Graphing Commands



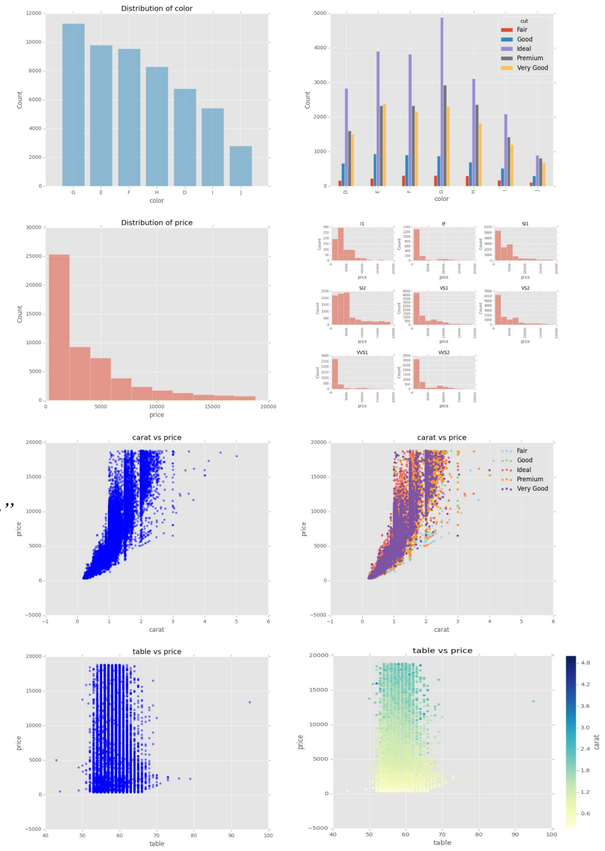
SPEAKING GRAPHS AND RESOLVING MISTRANSCRIPTION

1. *"bar chart of color"*
2. *"group by cut"*

1. *"show price"*
2. *"group by clarity"*

1. *"scatter carat and price"*
2. *"color by cut"*

1. *"plot table and price"*
2. *"color by carat"*



Target Type	Target	Candidate 1: Transcription	Candidate 2: Bigram Representation	Leven: Words	Leven: Meta(words)	Leven: MRC(words)	Jaro: Words	Jaro: Meta(words)	Jaro: MRC(words)
Single letter	*x*	*ex*	-	**1**	2	1	0	0	0
	x	*lacks*	-	**5**	2	4	0	0	0
	y	*why*	-	**2**	0	2	0	0	0
	y	*white*	-	**5**	2	3	0	0	0
One-word homophone	*carat*	*carrot*	-	**2**	0	1	0.822	1	0.917
	carat	*cart*	-	**1**	0	0	0.933	1	1
	depth	*debt*	-	**2**	2	2	0.783	0.556	0.722
	depth	*dead*	-	**3**	2	3	0.633	0.611	0.583
Two-word concatenation	*interestrate*	*interest rate*	*interestrate*	0	0	0	1	1	1
	interest_rate	*interest rate*	*interestrate*	1	0	1	0.974	1	0.822
Word and number	*user2016*	*user 2016*	*user2016*	0	0	0	1	1	1
	variable1	*variable one*	*variableone*	3	1	1	0.872	0.933	0.866
	variable1	*variable 1*	*variable1*	0	0	0	1	1	1
	under10	*under 10*	*under10*	0	0	0	1	1	1
	underten	*under 10*	*under10*	3	2	2	0.78	0.889	0.778
Syllable and number	*var1*	*bar one*	*barone*	4	2	2	0.611	0.611	0.556
Syllable and word	*quallife*	*quad life*	*quadlife*	1	1	1	0.917	0.917	0.833
Initialism and word	*osi model*	*osi model*	*osimodel*	1	1	0	0.963	0.944	1
Acronym and word	*sat score*	*sat score*	*satscore*	1	1	0	0.963	0.944	1

GRAMMAR OF GRAPHICS FOR SPEECH

Voice commands in this domain can be interpreted as being decomposable into at least two classes: statements indicating session-level actions, and statements indicating graph-level actions. This system uses hotword detection to identify a variety of session-level actions, like Quit, Reset, and Summarize, before processing graph-level details. Such a grouping enables a conversational approach, in which users build graphs over several steps.

```
class Graphic(object):
    """A general graph template.

    Declares all the attributes that a graphing library would need, to build
    the string used to plot the graph.
    """

    def __init__(self):
        """Defines characteristics of graph.

        Sets values for graph characteristics. Some are None, others are
        strings or numbers.
        """
        self.dataset = None
        self.filename = None
        self.data_cols = []
        self.grouping = None
        self.geom = None
        self.color = 'steelblue'
        self.xscale = [None, None]
        self.yscale = [None, None]
        self.xlab = None
        self.ylab = None
        self.title = None
        self.base = False
        self.add_smooth = False
        self.valid_graph = False
```

```
# Run speech recognition and graphing in a streaming format.
while 1:
    raw_input('Tap ENTER to continue.')
    with mic as source:
        audio = get_audio(r, source)
        try:
            text = r.recognize(audio)
            print('You said: ' + text)
        except LookupError:
            print("Didn't get audio.")
            continue
        # See if command is quit, save, reset, or edit.
        if text:
            terms = tokenize(text)

            # Decide what the terms indicate, and do the actions.
            if is_quit(terms):
                print 'Goodbye'
                return None
            elif is_reset(terms):
                plt.clf()
                g = copy(g_data_only)
                data_preview(g)
                print 'DEFINE a new graph.'
            elif is_summary(terms):
                data_preview(g)
                g.summarize()
            elif g.has_base():
                g = update_graph(g, terms)
                g = graph_if_valid(g, g_data_only)
            else:
                g = create_graph(g, terms)
                g = graph_if_valid(g, g_data_only)
```