Microsoft Research
FacultySummit

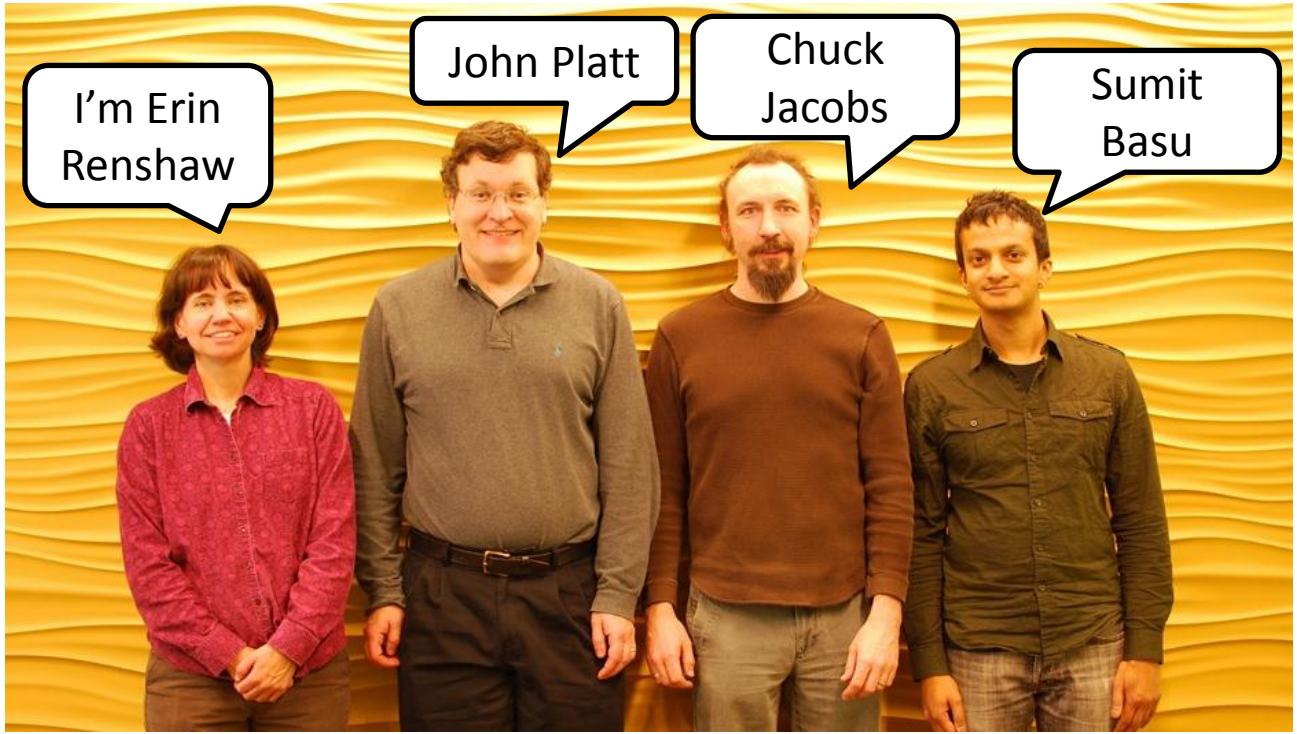
Part I: An Overview of Sho

Sumit Basu

FUTURE/WORLD 2011 2031

an introduction

The Sho Team

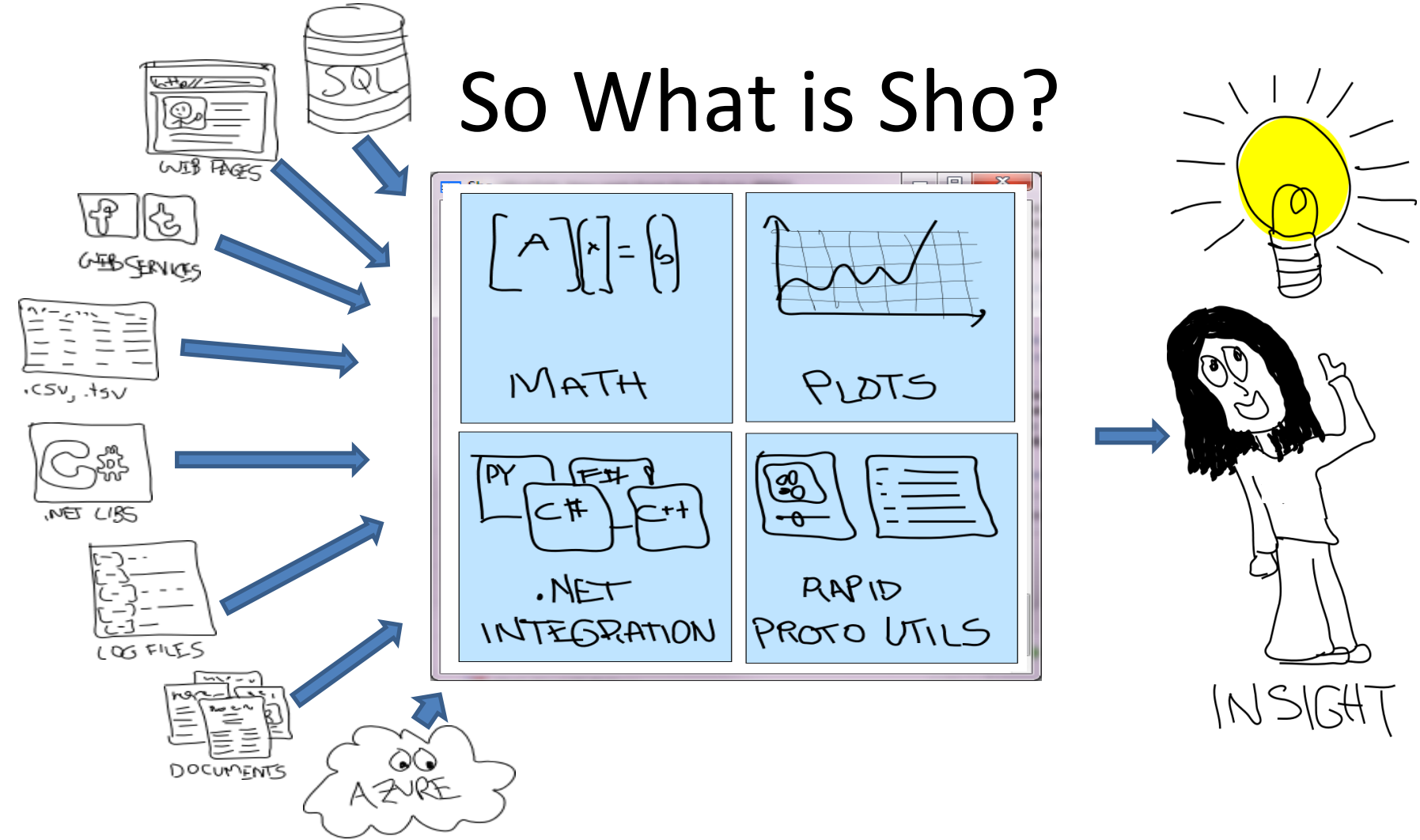


…plus a great deal of help from the fabulous folks of Developer Division

First, a Poll

- Who has used Matlab?
- Who has used Python?
- Who has used R?
- How many have wanted to (or had to) connect these to compiled libraries?

So What is Sho?



SHO DEMO

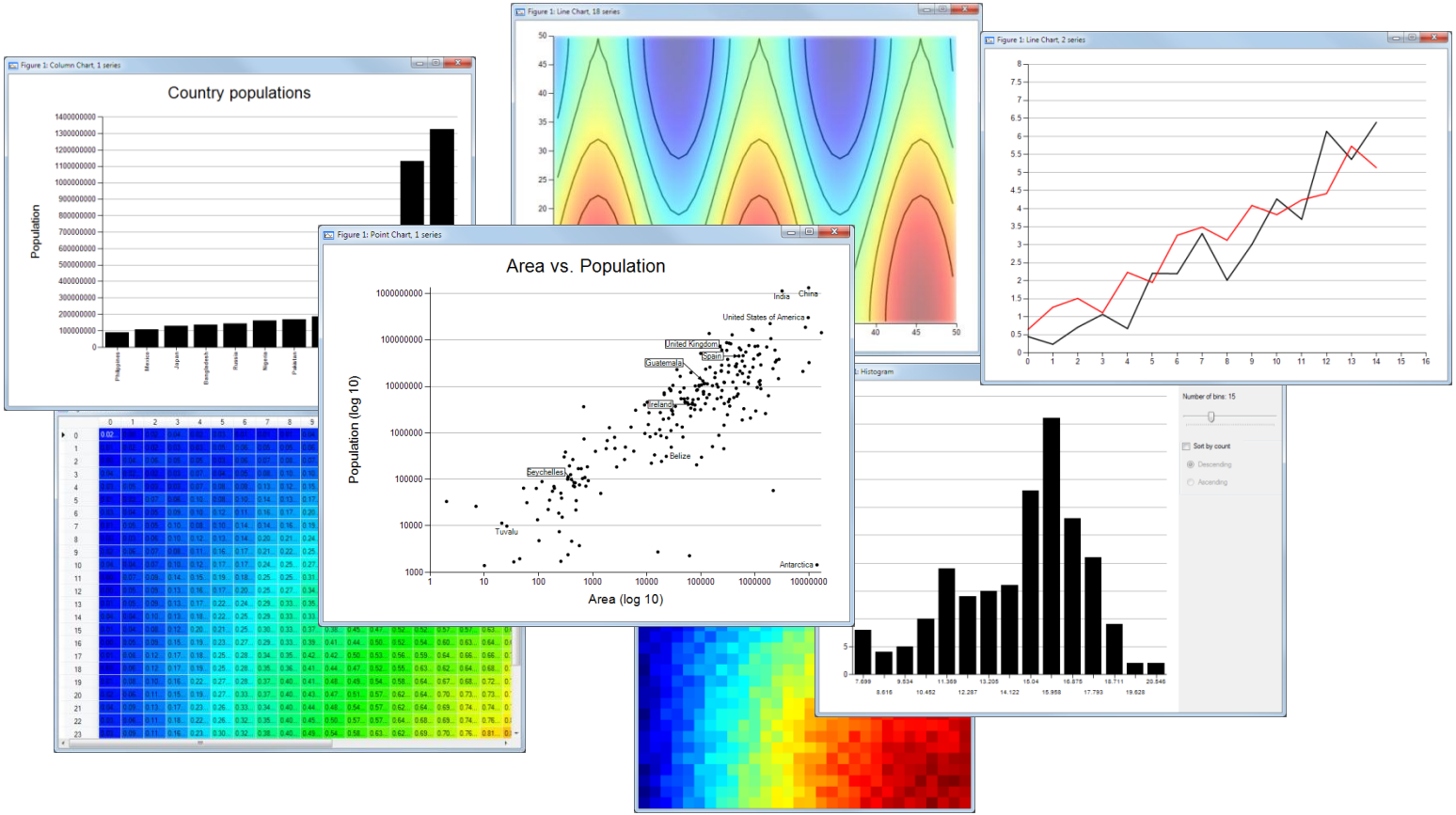
Matrix Math and Numeric Libraries

$$\begin{bmatrix} \hat{\mathbf{x}}_{t|t} \\ \hat{\mathbf{x}}_{t-1|t} \\ \vdots \\ \hat{\mathbf{x}}_{t-N+1|t} \end{bmatrix} = \begin{bmatrix} I \\ 0 \\ \vdots \\ 0 \end{bmatrix} \hat{\mathbf{x}}_{t|t-1} + \begin{bmatrix} 0 & \dots & 0 \\ I & 0 & \vdots \\ \vdots & \ddots & \vdots \\ 0 & \dots & I \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_{t-1|t-1} \\ \hat{\mathbf{x}}_{t-2|t-1} \\ \vdots \\ \hat{\mathbf{x}}_{t-N|t-1} \end{bmatrix} + \begin{bmatrix} K^{(1)} \\ K^{(2)} \\ \vdots \\ K^{(N)} \end{bmatrix} y_{t|t-1}$$

from http://en.wikipedia.org/wiki/Kalman_filter

- Sho Arrays
  - Dense and sparse arrays, from many types
  - Data Types: Double, Float, Int, Boolean, Object
  - Compact, math-like syntax: `x = inv(A.T*A)*A.T*b`
  - Operations on submatrices: `A[:,3] = b`
  - **FAST**: many operations speeded up by MKL
- Classes for common decompositions
  - SVD, LU, QR, Cholesky, Schur, Eigen

Visualization

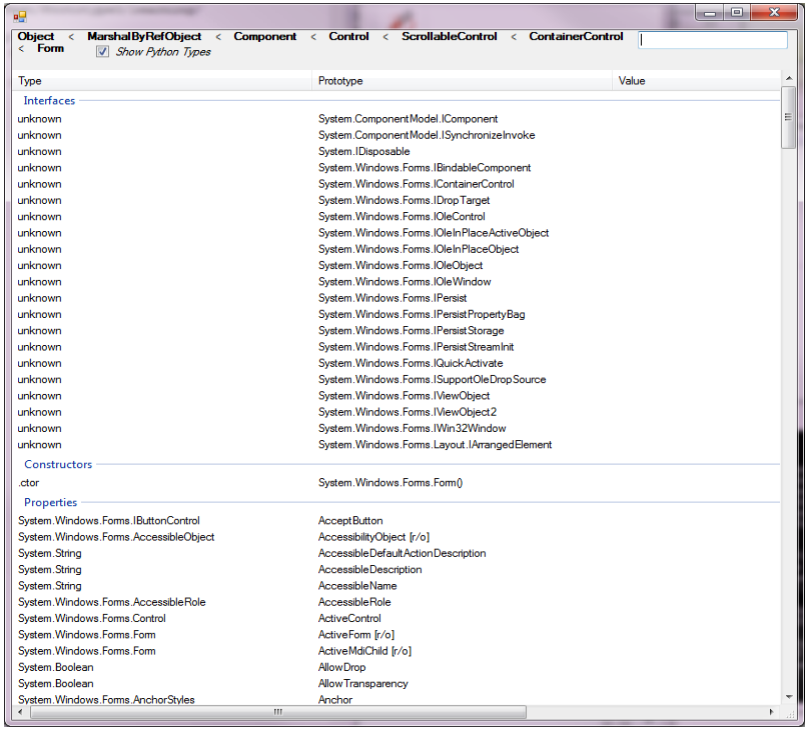


Console Features

- Tab Completion and Intellisense

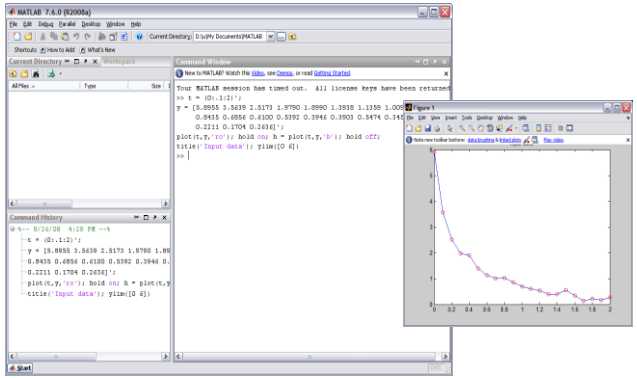
```
>>> Forms.Fo<TAB>
FolderBrowserDialog
FontDialog
Form
FormBorderStyle
FormClosedEventArgs
FormClosedEventHandler
FormClosingEventArgs
FormClosingEventHandler
FormCollection
...
```

- Doc



How is this Different from Matlab, R, etc.?

Typical environment



math & visualization

MSFT data stack

- Powerful datastructures
- .NET libraries
- Great languages/IDEs
- Excel, SQL, Sharepoint
- LINQ and PLINQ
- Web and cloud access

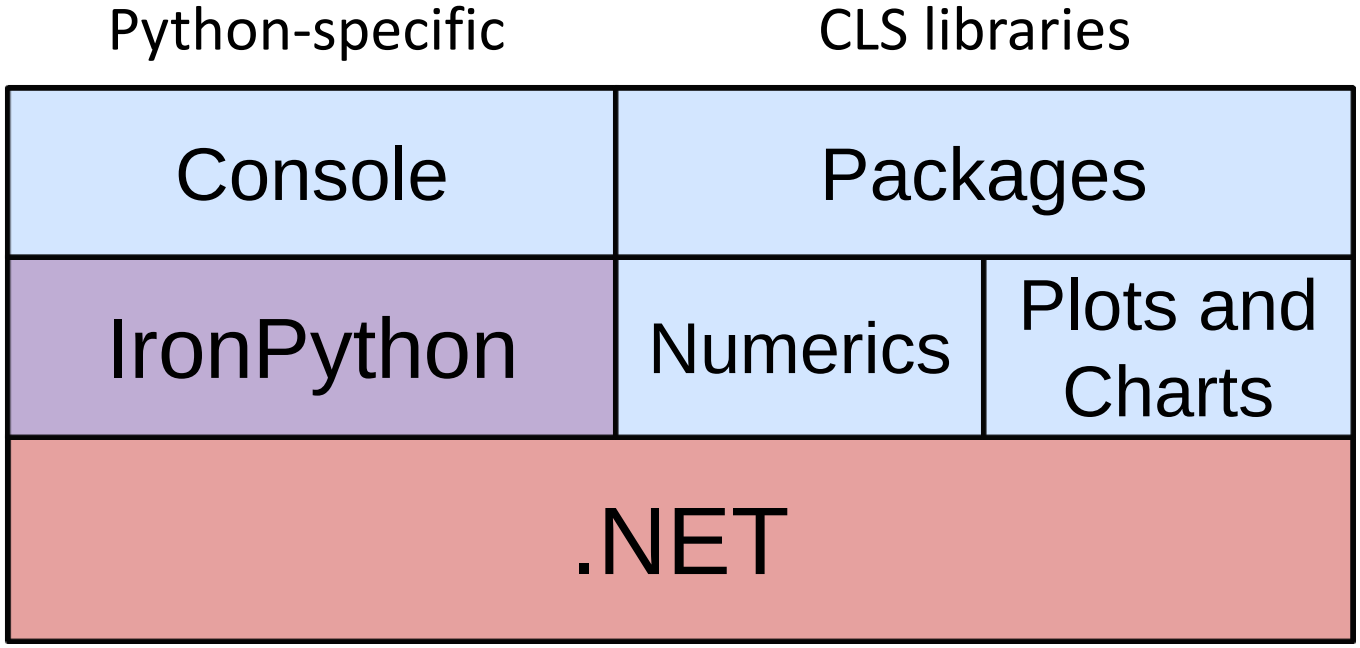
Algorithmic Folks

have this
want this

want this
have this

Prototypers

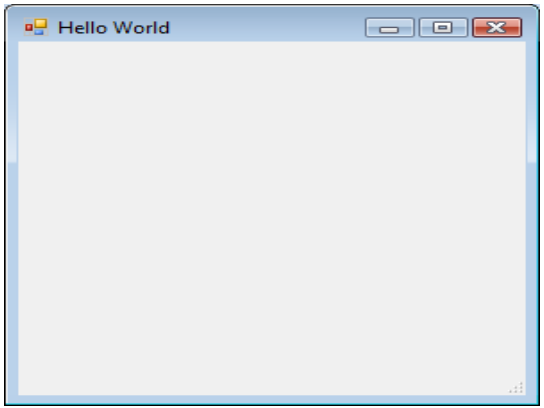
The Architecture of Sho



Sho is .NET All the Way Down

- IronPython talks directly to .NET
  - Instantiate arbitrary .NET objects
  - Subclass objects and override behaviors
  - Use interfaces, enumerators, etc.
  - No wrapping or decoration needed
- Example: Making a Form

```
>>> f = System.Windows.Forms.Form()
>>> f.Text = “Hello World”
>>> f.ShowDialog()
```



Using a .NET Library from Sho

```
>>>load('C:\\Windows\\...\\System.Speech.dll')
<Assembly System.Speech, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35>
```

load the speech DLL

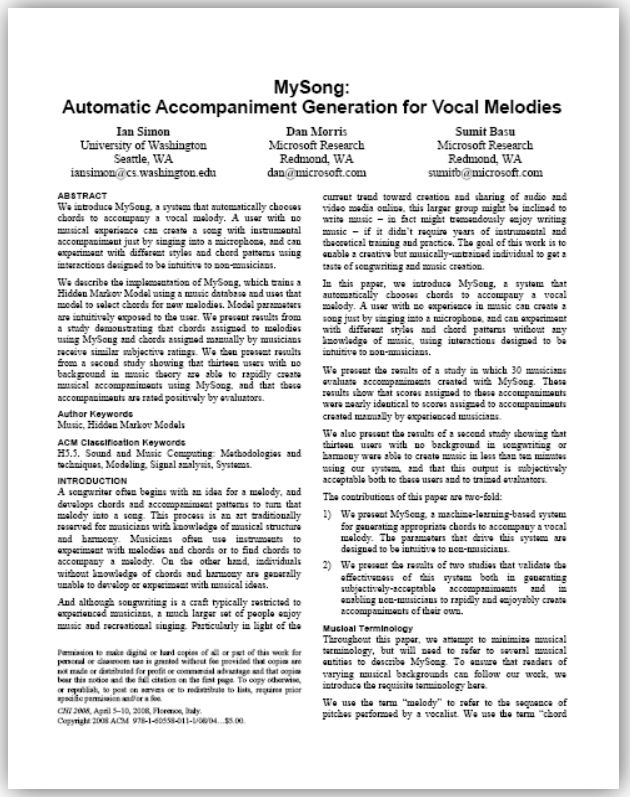
```
>>> from System.Speech.Recognition import
SpeechRecognitionEngine, DictationGrammar
>>> sre = SpeechRecognitionEngine()
>>> sre.SetInputToDefaultAudioDevice()
>>> sre.LoadGrammar(DictationGrammar())
>>> res = sre.Recognize()
```

set up the recognizer

the quick brown fox jumped over the lazy dog

```
>>> res.Text
'the quick brown fox jumped over the least got when'
```

Using a C# File Directly from Sho



MySong:
Automatic Accompaniment Generation for Vocal Melodies

Ian Simon
University of Washington
Seattle, WA
iansimon@cs.washington.edu

Dan Morris
Microsoft Research
Redmond, WA
dan@microsoft.com

Sumit Basu
Microsoft Research
Redmond, WA
sumitb@microsoft.com

ABSTRACT

We introduce MySong, a system that automatically chooses chords to accompany a vocal melody. A user with no musical experience can create a song with instrumental accompaniment just by singing into a microphone, and can experiment with different styles and chord patterns using interactions designed to be intuitive to non-musicians.

We describe the implementation of MySong, which trains a Hidden Markov Model using a music database and uses that model to select chords for new melodies. Model parameters are intuitively exposed to the user. We present results from a study demonstrating that chords assigned to melodies using MySong and chords assigned manually by musicians receive similar subjective ratings. We then present results from a second study showing that thirteen users with no background in music theory are able to rapidly create musical accompaniments using MySong, and that these accompaniments are rated positively by evaluators.

Author Keywords
Music, Hidden Markov Models

ACM Classification Keywords
H5.5. Sound and Music Computing: Methodologies and techniques, Modeling, Signal analysis, Systems.

INTRODUCTION

A songwriter often begins with an idea for a melody, and develops chords and accompaniment patterns to turn that melody into a song. This process is an art traditionally reserved for musicians with knowledge of musical structure and harmony. Musicians often use instruments to experiment with melodies and chords or to find chords to accompany a melody. On the other hand, individuals without knowledge of chords and harmony are generally unable to develop or experiment with musical ideas.

And although songwriting is a craft typically restricted to experienced musicians, a much larger set of people enjoy music and recreational singing. Particularly in light of the current trend toward creation and sharing of audio and video media online, this larger group might be inclined to write music – in fact might tremendously enjoy writing music – if it didn't require years of instrumental and theoretical training and practice. The goal of this work is to enable a creative but musically-untrained individual to get a taste of songwriting and music creation.

In this paper, we introduce MySong, a system that automatically chooses chords to accompany a vocal melody. A user with no experience in music can create a song just by singing into a microphone, and can experiment with different styles and chord patterns without any knowledge of music, using interactions designed to be intuitive to non-musicians.

We present the results of a study in which 30 musicians evaluate accompaniments created with MySong. These results show that scores assigned to these accompaniments were nearly identical to scores assigned to accompaniments created manually by experienced musicians.

We also present the results of a second study showing that thirteen users with no background in songwriting or harmony were able to create music in less than ten minutes using our system, and that this output is subjectively acceptable both to these users and to trained evaluators.

The contributions of this paper are two-fold:

1) We present MySong, a machine-learning-based system for generating appropriate chords to accompany a vocal melody. The parameters that drive this system are designed to be intuitive to non-musicians.
2) We present the results of two studies that validate the effectiveness of this system both in generating subjectively-acceptable accompaniments and in enabling non-musicians to rapidly and enjoyably create accompaniments of their own.

Musical Terminology

Throughout this paper, we attempt to minimize musical terminology, but will need to refer to several musical entities to describe MySong. To ensure that readers of varying musical backgrounds can follow our work, we introduce the requisite terminology here.

We use the term "melody" to refer to the sequence of pitches performed by a vocalist. We use the term "chord

CHI 2008, April 5–10, 2008, Florence, Italy.


```
>>> load("IFilterReader.cs")
>>> import IFilter
>>> txt =
    IFilter.DefaultParser.Extract
    ("MySong.pdf")
>>> print txt[:1000]
    MySong: Automatic Accompaniment
    Generation for Vocal Melodies
    Ian Simon University of
    Washington Seattle, WA
    iansimon@cs.washington.edu Dan
    Morris Microsoft Research
    Redmond, WA dan@microsoft.com
    Sumit Basu Microsoft Research
    Redmond, WA
    sumitb@microsoft.com ABSTRACT
```

Using a Web Service from Sho

```
c:\src\sho\playpen>"c:\Program Files\Microsoft SDKs\Windows\v6.0A\Bin\wsdl.exe"
http://api.search.live.net/search.wsdl?AppID=XXXXXXXX /namespace:livesearch
Writing file 'c:\src\sho\playpen\LiveSearchService.cs'.
```

get the web service API

```
>>> load("LiveSearchService.cs")
>>> from livesearch import LiveSearchService, SearchRequest, SourceType
>>> sc = LiveSearchService()
>>> r = SearchRequest()
>>> r.AppId = "XXXXXXXX"
>>> r.Sources = System.Array.CreateInstance(SourceType, 1)
>>> r.Sources[0] = SourceType.Web
>>> r.Query = "Internships at Microsoft Research"
>>> response = sc.Search(r)
>>> for elt in response.Web.Results:
        print "["+elt.Title+"]", elt.Description, '\n',elt.Url,'\n'
```

set up a query

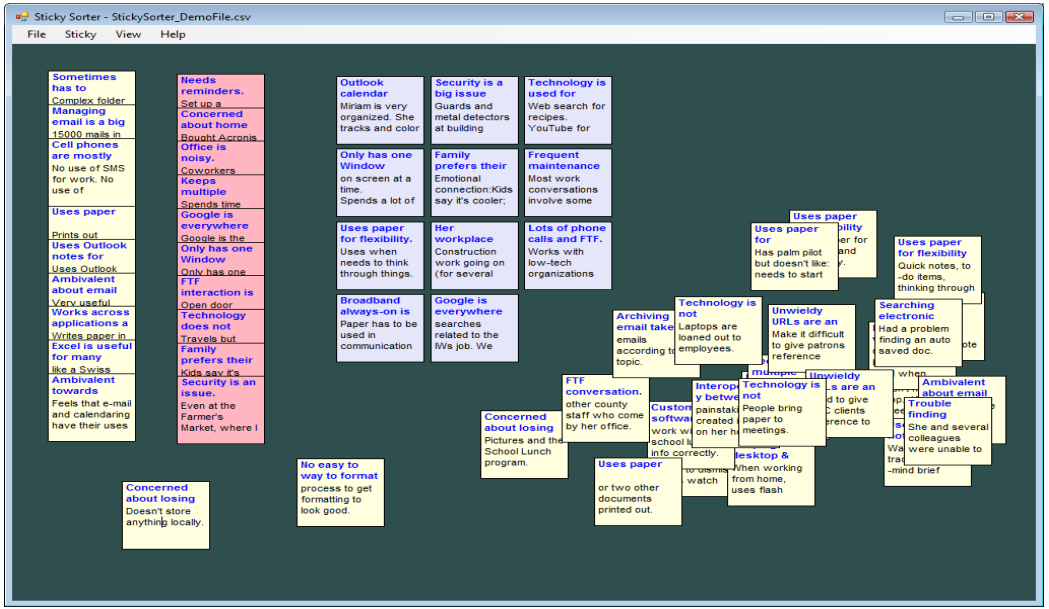
print the results

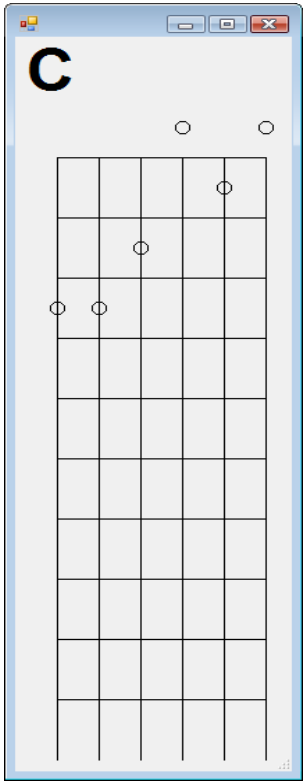
[Apply for an Internship - Microsoft Research] Intern applicants with strong academic achievement in fields such as Computer Science, Electrical Engineering, Math, or Social Sciences with a focus on technology are preferred.
http://research.microsoft.com/en-us/jobs/intern/apply.aspx

bing

Easy to build GUIs

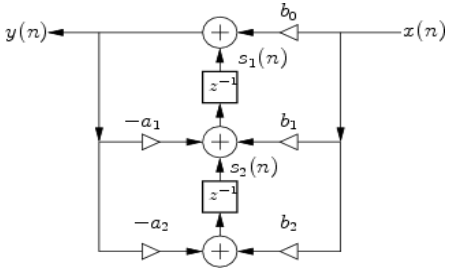
- Create WinForms GUIs in Sho
- Use GUIs built in Visual Studio
- WPF windows/applications/XAML

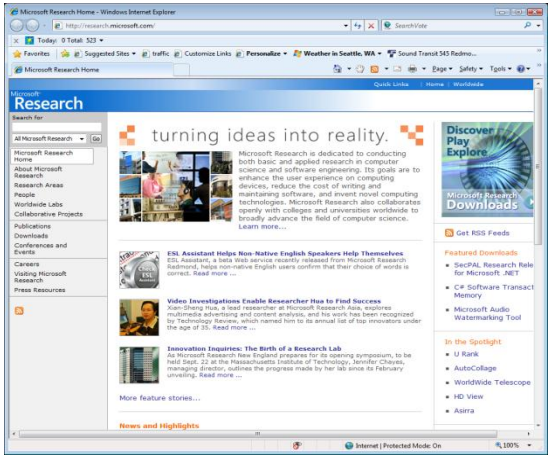


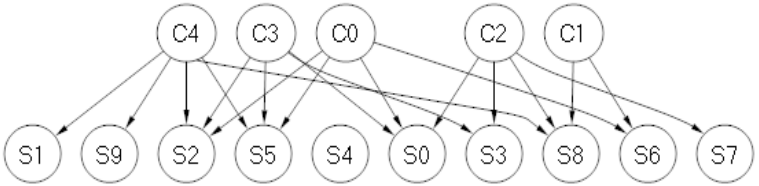


Sho Packages

- Released
  - Statistics/Random Number Generation
  - Optimization
  - Signal Processing
  - Database (SQL) Access
  - Cluster Computing (HPC)
  - Cloud Computing (Azure)
- To Be Released Soon
  - ShoCGI (dynamic web apps)
  - Kinect
- Under Development
  - Machine Learning
  - Image Processing
  - Graph Layout

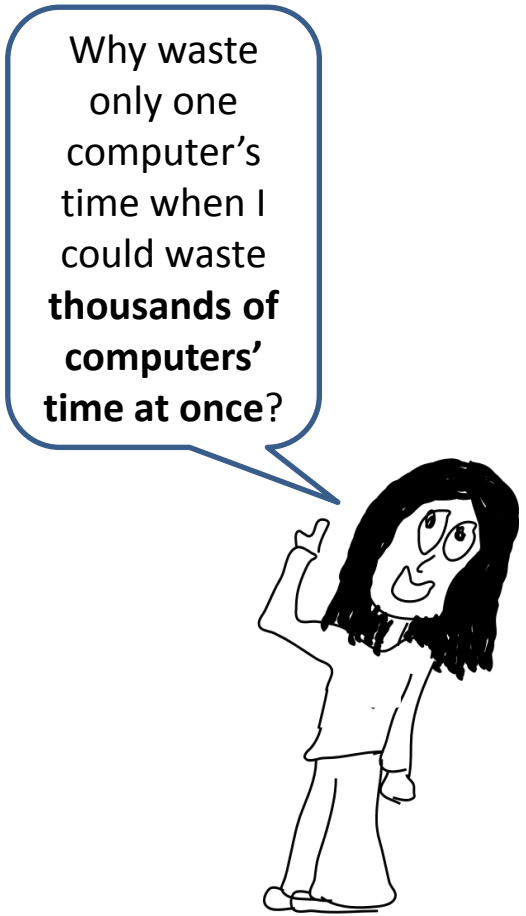






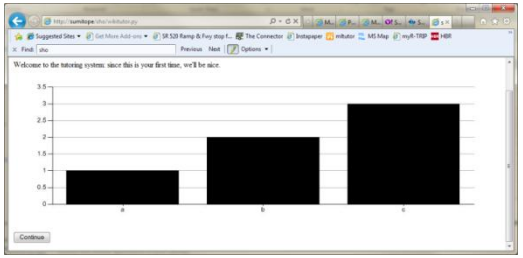
HPC and Azure

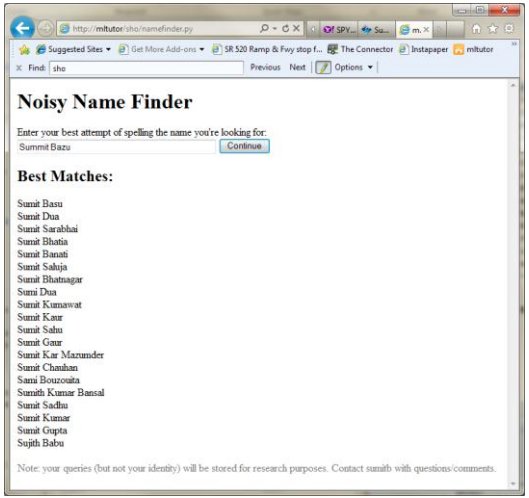
```
# original function with one parameter value and data
directory
>>> res = basisboost.clusterproc( (15,2,1), 0.05,
“c:/researchjournal/briefcase/sentimentdata/cluster”)

# set up cluster
>>> cl = HPC.clustersetup("RR1-N13-09-H44",
"//MSR-ARRAYS/SCRATCH/MSR-POOL/09-H44/redmond/sumitb")

# run on cluster
>>> session = HPC.clusterrun(cl,   basisboost.clusterproc,
(15,2,1),
basisboost.createseq( 20*list(drange(0.05,0.8,0.05))),
"c:/researchjournal/briefcase/sentimentdata/cluster")

# get results
>>> clusterresults = session.getresults()
```



ShoCGI

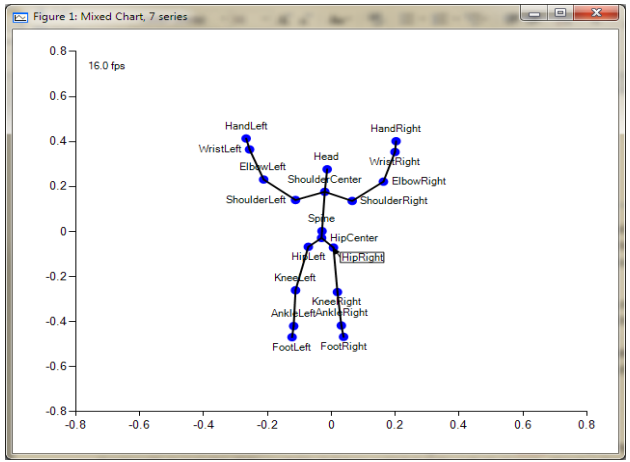
- Create dynamic web applications using your Sho code with a minimum of extra work
- Cookie support and persistent session store
- Include Sho Plots in your pages
- Create multi-page applications with logins, etc., in a single Python file in less than a hundred lines of code

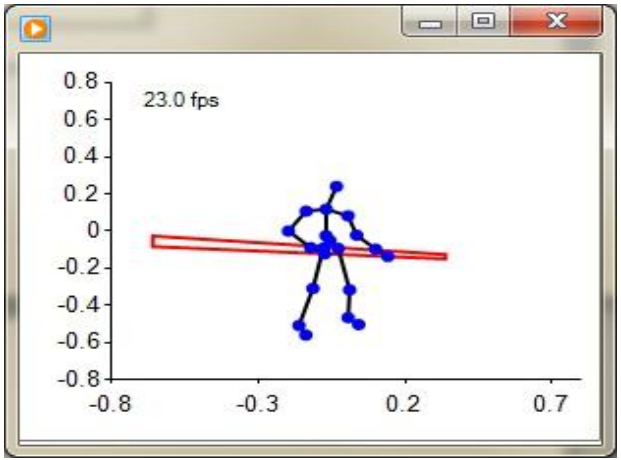




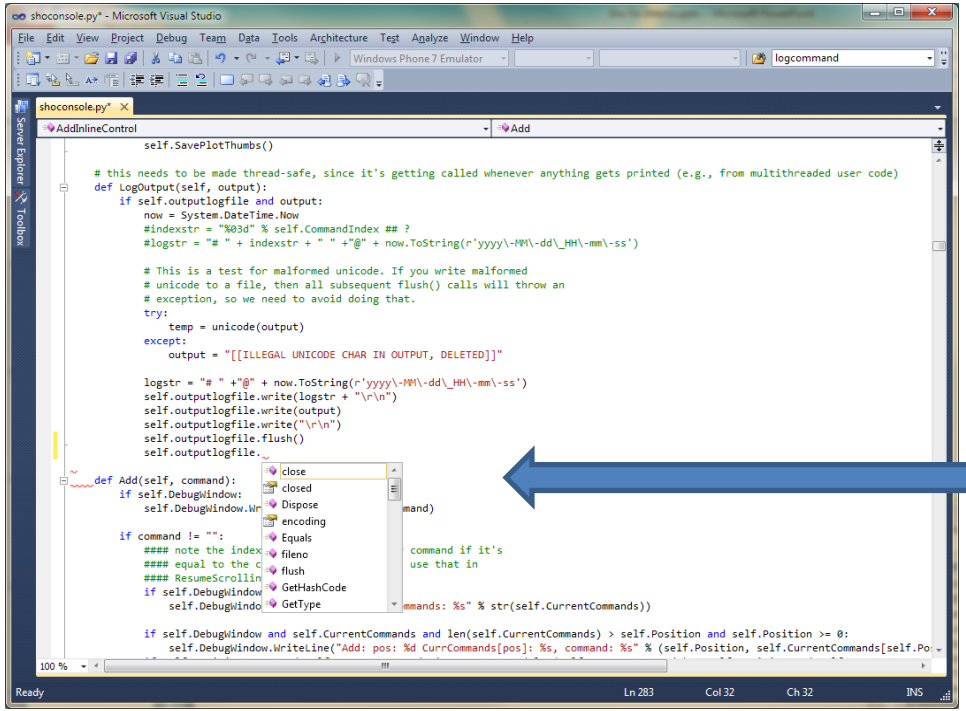
Kinect for Sho

- Based MSR's KSDK (to be released this spring)
- Get real-time skeleton data
- Use Sho viz tools to make real-time, animated skeletons in about 50 lines of code





Sho and Visual Studio



Python Tools for Visual Studio

- Syntax highlighting
- Smart indenting
- Intellisense
- Go to definition
- Find all references
- Cross-language debugging
- Inspect python and .net values

PYTHON TOOLS FOR VISUAL STUDIO DEMO

Using Sho Libraries from C#

```
using ShoNS.Array;
using ShoNS.Visualization;
using ShoNS.MathFunc;
namespace CSharpExample
{
    class Program
    {
        static void Main(string[] args)
        {
            System.Environment.SetEnvironmentVariable("LALDIR",
               @"C:/Program Files/Sho",
               System.EnvironmentVariableTarget.Process);
            // Create an array to work with
            DoubleArray x = ArrayRandom.RandomDoubleArray(10, 10);
            x = x * x.T;
            SVD svd = new SVD(x);
            // Create a figure and display a barchart of the
            // logs of the singular values
            IFigure f1 = ShoPlotHelper.Figure();
            DoubleArray logVals = ArrayMath.Log(svd.D.Diagonal + 1.0);
            f1.Bar(logVals);
        }
    }
}
```

include the libs

a line of setup

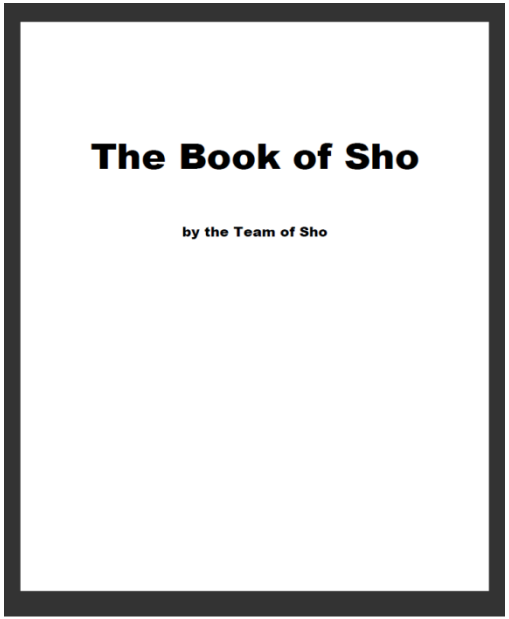
make an array

do some math

show a plot

How Do I Get Started?

- **The Site/Blog/Forum/etc.:** http://research.microsoft.com/sho
- **The Book:**
  - In PDF and HTML
- **The API Reference:**
  - MSDN-style docs for C#



Take-Home Messages

- Sho has **linear algebra**, **visualization**, and **packages**, both in the **REPL** and from **C#**
- Sho lets you **move smoothly between script and compiled code**
- Sho lets you use **existing .NET libraries**

http://research.microsoft.com/sho

Microsoft Research
FacultySummit

Part II: Sho Workshop

Sumit Basu and Jan Vitek

FUTURE/WORLD 2011 2031

Format and Logistics

- Install Sho and PTVS
  - http://research.microsoft.com/sho
  - http://pytools.codeplex.com (for editing/debugging)
- Option 1:
  - Pick one of the following small exercises
  - Join up with others interested in the same problem
  - Jan and I will wander around helping where needed
- Option 2:
  - We can walk through the problems together
- Code and data for exercises is at http://research.microsoft.com/~sumitb/shoexercises.zip

EXERCISES

1: Plotting Data

- Plot country populations and areas
- Materials:
  - CSV file containing country names, populations, areas
- Goal:
  - Plot areas vs. populations
  - Add names as labels
  - Fit a line/curve to the data and plot it

2: Text Analysis

- Frequency analysis of text files
- Materials
  - Small corpus of text files (paper abstracts)
- Goals
  - Compute frequency of all words
  - Create bar chart of N most frequent words

3: Labeling Tool (GUI)

- Create a labeling tool for labeling sentence sentiment
- Materials
  - File containing sentences to be labeled
- Goal
  - Simple GUI for labeling

4: Voice Calculator

- Do simple math with voice I/O
- Materials:
  - Speech recognition and synthesis example loop
- Goal
  - Make audio-only calculator allowing commands like add, multiply, subtract, divide

Microsoft Research
FacultySummit

FUTURE/WORLD

2011 — 2031