Developing and Deploying Intelligent Chat Bots

Microsoft

Bots with Cognitive Services API

Objectives

The aim of this lab is to build a bot called SentiBot that integrates Text Analytics Cognitive Services API. Specifically, we will use the Sentiment Analysis feature of Text Analytics Cognitive Services API to evaluate the sentiment of any message sent to the bot.

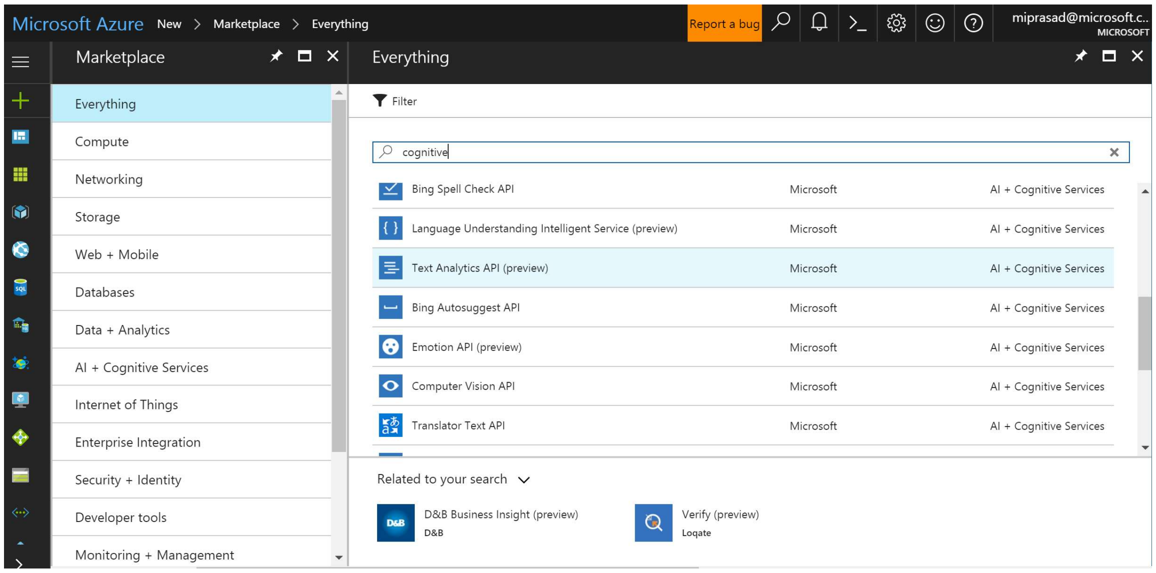
The lab is adapted from:

https://docs.botframework.com/en-us/bot-intelligence/language/#example-emotional-bot

.

Setup

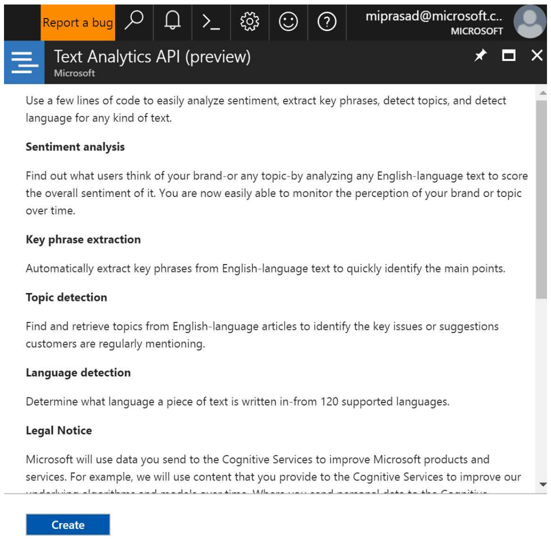
1. Text Analytics API Key:
    a. Go to portal.azure.com and sign in. Sign up for an account if you do not have one.
    b. Select the Text Analytics API as shown below:



cognitive

Bing Spell Check API	Microsoft	AI + Cognitive Services
Language Understanding Intelligent Service (preview)	Microsoft	AI + Cognitive Services
Text Analytics API (preview)	Microsoft	AI + Cognitive Services
Bing Autosuggest API	Microsoft	AI + Cognitive Services
Emotion API (preview)	Microsoft	AI + Cognitive Services
Computer Vision API	Microsoft	AI + Cognitive Services
Translator Text API	Microsoft	AI + Cognitive Services

    c. Create an account by filling out all the mandatory fields after clicking "create". Copy the key from the blade after you successfully create.



Text Analytics API (preview)
Microsoft

Use a few lines of code to easily analyze sentiment, extract key phrases, detect topics, and detect language for any kind of text.

Sentiment analysis

Find out what users think of your brand-or any topic-by analyzing any English-language text to score the overall sentiment of it. You are now easily able to monitor the perception of your brand or topic over time.

Key phrase extraction

Automatically extract key phrases from English-language text to quickly identify the main points.

Topic detection

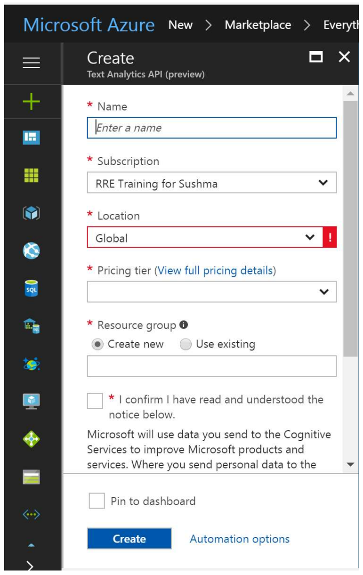
Find and retrieve topics from English-language articles to identify the key issues or suggestions customers are regularly mentioning.

Language detection

Determine what language a piece of text is written in-from 120 supported languages.

Legal Notice

Microsoft will use data you send to the Cognitive Services to improve Microsoft products and services. For example, we will use content that you provide to the Cognitive Services to improve our



Create
Text Analytics API (preview)

* Name
Enter a name

* Subscription
RRE Training for Sushma

* Location
Global

* Pricing tier (View full pricing details)

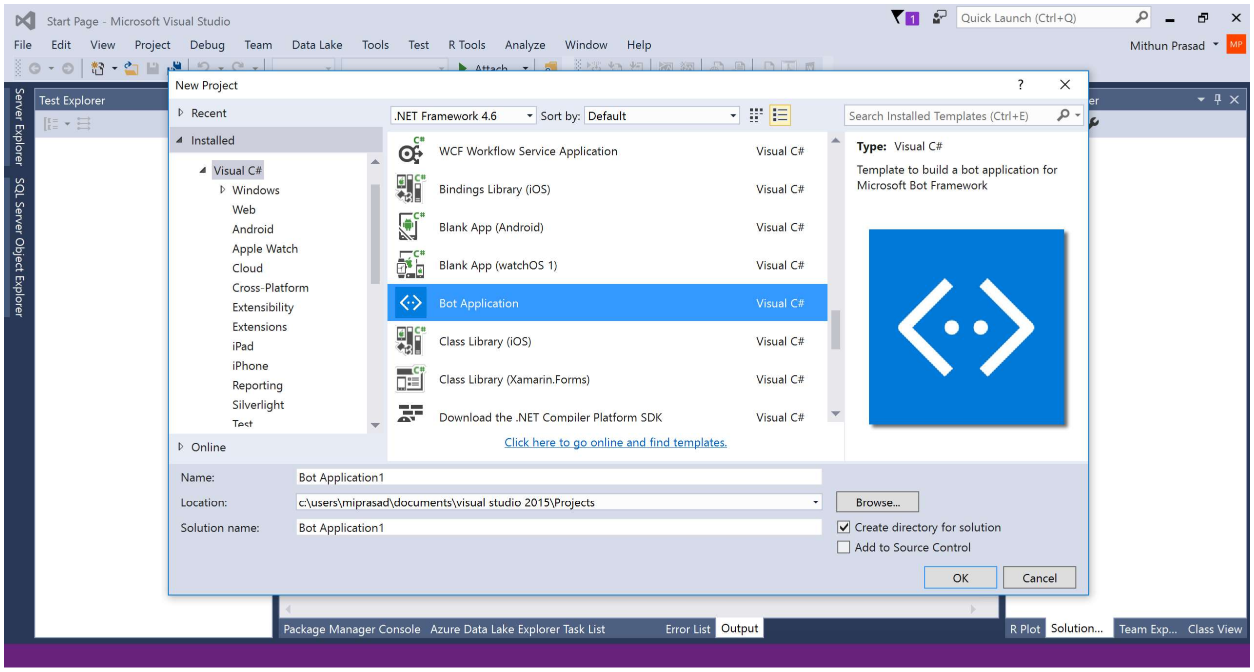
* Resource group
(● Create new) ○ Use existing

☐ * I confirm I have read and understood the notice below.

Microsoft will use data you send to the Cognitive Services to improve Microsoft products and services. Where you send personal data to the

☐ Pin to dashboard

2. Create a new Bot Project using Bot Application Template in Visual Studio as shown below:



New Project

- Recent
- Installed
  - Visual C#
    - Windows
    - Web
    - Android
    - Apple Watch
    - Cloud
    - Cross-Platform
    - Extensibility
    - Extensions
    - iPad
    - iPhone
    - Reporting
    - Silverlight
    - Test
- Online

.NET Framework 4.6 Sort by: Default

Search Installed Templates (Ctrl+E)

WCF Workflow Service Application	Visual C#
Bindings Library (iOS)	Visual C#
Blank App (Android)	Visual C#
Blank App (watchOS 1)	Visual C#
Bot Application	Visual C#
Class Library (iOS)	Visual C#
Class Library (Xamarin.Forms)	Visual C#
Download the .NET Compiler Platform SDK	Visual C#

Click here to go online and find templates.

Type: Visual C#

Template to build a bot application for Microsoft Bot Framework

Name: Bot Application1
Location: c:\users\miprasad\documents\visual studio 2015\Projects
Solution name: Bot Application1

☑ Create directory for solution
☐ Add to Source Control

SentiBot

Create a new C# class file (Sentiment.cs) with the following code. The class will serve as our model for the JSON input/output of the Text Analytics API.

```
namespace TextSentiBot
{
    // Classes to store the input for the sentiment API call
    public class BatchInput
    {
        public List<DocumentInput> documents { get; set; }
    }
    public class DocumentInput
    {
        public double id { get; set; }
        public string text { get; set; }
    }

    // Classes to store the result from the sentiment analysis
    public class BatchResult
    {
        public List<DocumentResult> documents { get; set; }
    }
    public class DocumentResult
    {
        public double score { get; set; }
        public string id { get; set; }
    }
}
```

Next, go to MessagesController.cs and add the following namespaces if needed.

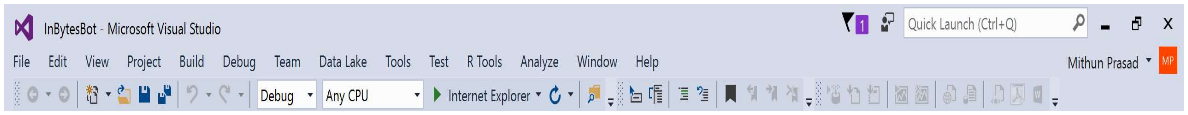
```
using System;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Threading.Tasks;
using System.Web.Http;
using System.Web.Http.Description;
using Microsoft.Bot.Connector;
using Newtonsoft.Json;
using System.Collections.Generic;
using System.Net.Http.Headers;
using System.Text;
```

The code receives the user message, calls the sentiment analysis endpoint and responds accordingly to the user. Replace the Post method in MessageController.cs with the following code.

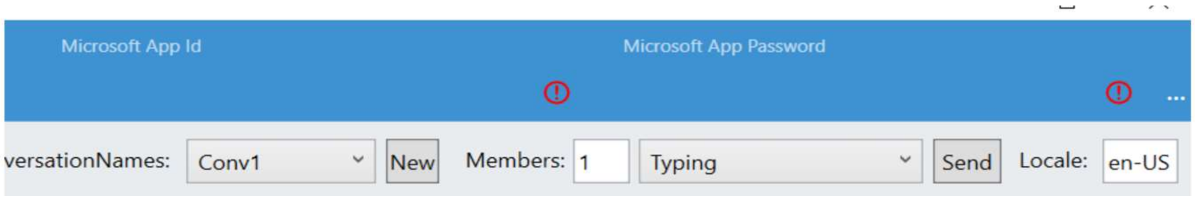
The apiKey in the below code would need to be replaced with your key.

```
        public async Task<HttpResponseMessage> Post([FromBody]Activity activity)
        {
            ConnectorClient connector = new ConnectorClient(new
            Uri(activity.ServiceUrl));

            if (activity == null || activity.GetActivityType() !=
            ActivityTypes.Message)
            {
                //add code to handle errors, or non-messaging activities
            }

            const string apiKey = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
            string queryUri =
"https://westus.api.cognitive.microsoft.com/text/analytics/v2.0/sentiment";

            HttpClient client = new HttpClient();
            client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", apiKey);
            client.DefaultRequestHeaders.Add("Accept", "application/json");
            BatchInput sentimentInput = new BatchInput();

            sentimentInput.documents = new List<DocumentInput>();
            sentimentInput.documents.Add(new DocumentInput()
            {
                id = 1,
                text = activity.Text
            });

            var sentimentJsonInput = JsonConvert.SerializeObject(sentimentInput);
            byte[] byteData = Encoding.UTF8.GetBytes(sentimentJsonInput);
            var content = new ByteArrayContent(byteData);
            content.Headers.ContentType = new
            MediaTypeHeaderValue("application/json");
            var sentimentPost = await client.PostAsync(queryUri, content);
            var sentimentRawResponse = await
            sentimentPost.Content.ReadAsStringAsync();
            var sentimentJsonResponse =
            JsonConvert.DeserializeObject<BatchResult>(sentimentRawResponse);
            double sentimentScore = sentimentJsonResponse.documents[0].score;

            var replyMessage = activity.CreateReply();
            replyMessage.Recipient = activity.From;
            replyMessage.Type = ActivityTypes.Message;

            if (sentimentScore > 0.9)
            {
                replyMessage.Text = $"This appears quite positive to me.";
            }
            else if (sentimentScore < 0.1)
            {
                replyMessage.Text = $"This appears quite negative to me.";
            }
            else
            {
                replyMessage.Text = $"I can not decipher the sentiment here.  Please
                try again or add more information.";
            }

            await connector.Conversations.ReplyToActivityAsync(replyMessage);
            var response = Request.CreateResponse(HttpStatusCode.OK);
            return response;
        }
```

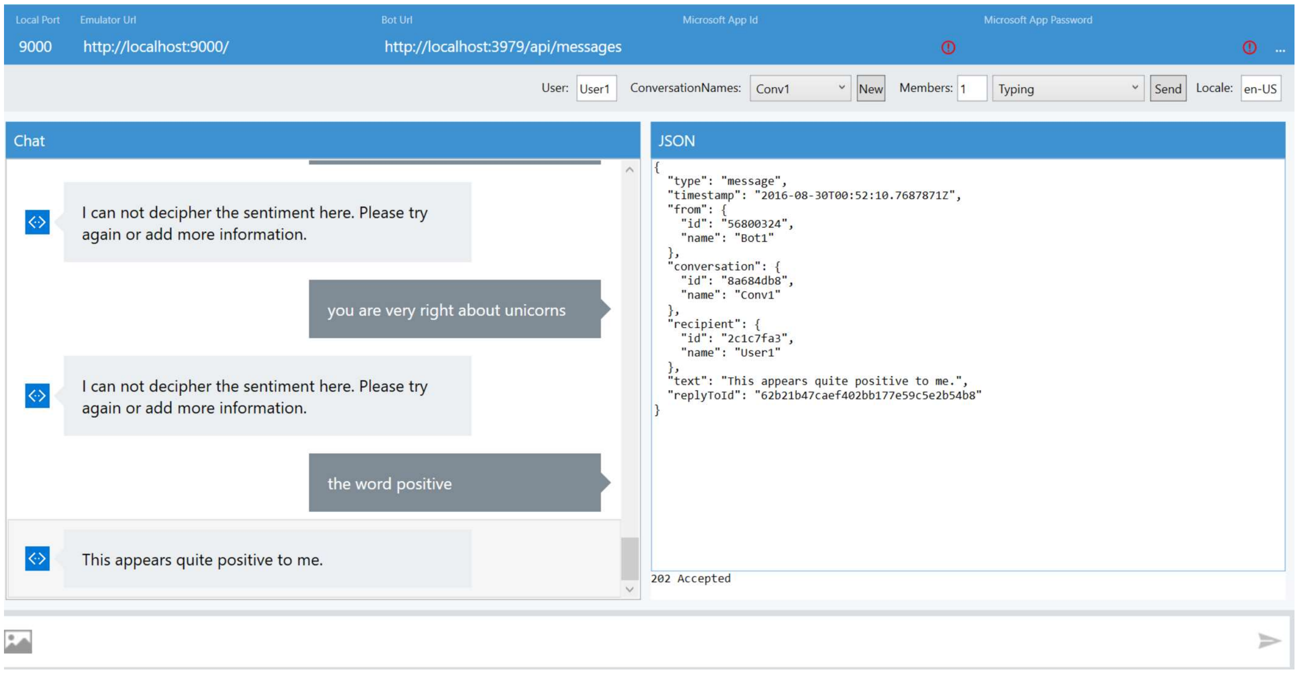
Run from Visual Studio by clicking *Internet Explorer* as shown below. This should open a browser window with the endpoint (showing the default.htm file from project).



Open the bot emulator and change the Bot Url to http://localhost:port/api/messages. You will need to add /api/messages to the Url. Ensure the fields Id and password fields are clear.



Test it in the emulator by sending very positive and very negative messages:



I can not decipher the sentiment here. Please try again or add more information.

you are very right about unicorns

I can not decipher the sentiment here. Please try again or add more information.

the word positive

This appears quite positive to me.

```
{
  "type": "message",
  "timestamp": "2016-08-30T00:52:10.7687871Z",
  "from": {
    "id": "56800324",
    "name": "Bot1"
  },
  "conversation": {
    "id": "8a684db8",
    "name": "Conv1"
  },
  "recipient": {
    "id": "2c1c7fa3",
    "name": "User1"
  },
  "text": "This appears quite positive to me.",
  "replyToId": "62b21b47caef402bb177e59c5e2b54b8"
}
```

```
202 Accepted
```

Exercise

1. You will find that the sentiment of the sentences has to be quite extreme for the app to decipher. Can you change the sentiment thresholds in the code for the app to be more sensitive?

2. Can you integrate the key phrase extraction API to extract keywords from any text sent to the bot?