Relatório de Progresso

Closer ChatBOT – VOC

Disciplina: Projeto e Seminário
Curso: Engenharia Informática e de Computadores

Arnaldo Tema
N.º aluno: 39342

Lisboa, 2 de Maio de 2017

Índice

Progresso

- **Envio e Armazenamento**
  Foi desenvolvido um serviço (C-Sharp) que, através de um FTP (File Transfere Protocol) e respetiva autenticação (User e Password), permite que sejam adicionados ficheiros de áudio numa diretoria existente numa máquina com um determinado IP.

- ***Standardização***
  Foi desenvolvido um serviço que, estando à escuta de novos ficheiros adicionados a uma certa diretoria (situada na pasta /upload), converte-os para ficheiros áudio .wav (para que seja possível processa-los com a plataforma SoX), guarda os ficheiros convertidos na pasta devida (pasta /download) e encaminha para a API de processamento. [Nota: de momento há um serviço para cada etapa de modo a que, no futuro, seja possível o processamento multi thread]

- **Otimização de áudio e Pré processamento**
  Foi iniciado o desenvolvimento da API de Processamento:
  Foi implementada uma API – ainda em desenvolvimento- que dá suporte a todas as funcionalidades SoX mais importantes, para que seja possível utilizar esta plataforma como se de uma aplicação consola se tratasse. Esta API permitirá o uso de funções como *trim, silence, noise reduction* e criação de perfis de ruído que será útil na API de Processamento final.

- **ChatBOT – Testes VOC**
  Através das APIs desenvolvidas, já foi possível utilizar as funções de *trim* de silêncio com índices introduzidos (por exemplo: retirar 5 segundos de silêncio e substitui-los por 1 segundo de silêncio) e de criação de perfis de ruído – com o propósito de, posteriormente, fazer um filtro de ruído por chamada, retira-lo, e melhorar a qualidade do sinal obtido – em chamadas reais de empresas de contact-center e testar o seu workflow.

  Nestes testes foi possível obter com sucesso os ficheiros convertidos de qualquer tipo de ficheiro áudio (excepto tipos raw) para ficheiros .wav nas pastas desejadas, voltar a captura-los para atribuir processamento, efetuar o *trim* de silêncio do mesmo e, em alguns casos, identificar um perfil de ruído.

Problemas

Perfil de Ruído (Noise Profile)
A criação manual de perfis de ruído é bastante trivial. Resume-se a analisar um determinado ficheiro de áudio e escolher um intervalo que consideremos ruído (2 segundos é suficiente).
No entanto, faze-lo programaticamente é bastante mais complexo.

Determinar, com base numa análise pormenorizada dum ficheiro de áudio, se um certo intervalo de tempo se trata de ruído ou som que se quer analisar é algo bastante complicado porque existem milhares de vozes cujas frequências diferem em grande escala, ficheiros de áudio cuja qualidade é questionável o que faz com que sempre que um dos intervenientes fale pareça apenas ruído e mais situações que ainda não foram testadas, mas certamente trarão outros inconvenientes.

- Dever-se-á criar um número estático de perfis de ruído e atribuí-los consoante as frequências de cada ficheiro de áudio?

- Dever-se-á tentar arranjar um algoritmo que permita capturar um intervalo de tempo em que seja ruído e criar um perfil para cada ficheiro?

Neste momento, grandes sistemas como a Microsoft, Google e o IOS tratam apenas de reduzir o ruído com base em Templates de som – resumidamente (e falando muito brutamente), se um ficheiro de áudio tem uma média de frequências de 10 unidades de frequência, então tudo o que for 2 é considerado ruído. Se, feiro esse filtro, se obtiver muito pouco áudio ou uma tradução para texto muito pobre em palavras, então o ficheiro é descartado e criado outro com outro perfil de ruído.

Acontece que, sabendo que o VOC e o AuditBOT se destinam inicialmente a empresas portuguesas, sabe-se que a probabilidade de os ficheiros virem com uma qualidade baixa é relativamente alta, portanto, regendo-nos pelas práticas das entidades mencionadas acima, o produto seria pouco eficaz.

Stereo – Mono
Para o desenvolvimento da componente de auditoria de chamadas (AuditBOT) será necessário descriminar e identificar com a maior clareza possível os diferentes intervenientes da chamada.
Ora, esta separação é direta quando se trata de um ficheiro de áudio stereo, pois os canais de áudio são gravados em separado logo torna-se possível essa divisão, no entanto, num ficheiro mono os fins de reprodução encontram-se misturados num só canal, o que faz com que seja necessária uma separação de áudios com base, novamente, nas frequências diferentes existentes no ficheiro. Isso traz consequências diretas em métodos da API de processamento, por

exemplo, quando múltiplos intervenientes falam em simultâneo – é perdido o critério de: Frequência X – voz X; Frequência Y – Voz Y.

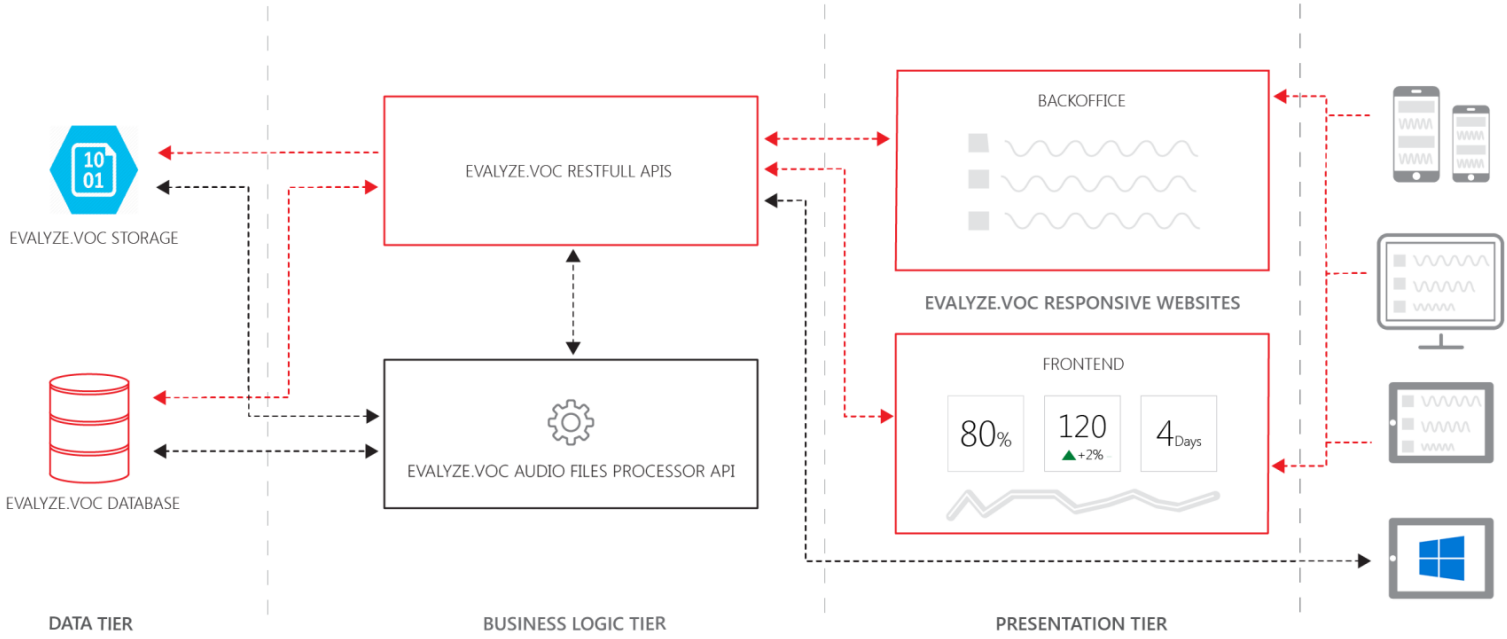
Dependência de APIs externas

Em aplicações que usam outras APIs para o seu correto funcionamento, fica sempre presente essa dependência em todas as versões lançadas, portanto, é necessário procurar que as APIS RESTFull usadas para Logging, Autenticação. DataBase querying e APIs externas para a tradução sejam o mais fidedignas possível.

Plano

Devido a uma análise mais aprofundada do projeto, foram feitas algumas atualizações no seu plano.

O projeto VOC (Voice of the Costumer – componente do ChatBOT) encontra-se distribuído em três camadas:



Camada de Dados (Data Tier)

Armazenamento:

Nesta camada, a propósito de *storage* o VOC irá contar com o *Azure* Storage, *cloud-based*, a fim de:

- Todos os ficheiros de áudio (os ficheiros originais e os processados) serão armazenados nesta componente como objetos (blobs) sob a forma de instância Blob.
- Adicionalmente, toda a informação Log proveniente da aplicação será armazenada nesta componente sob a forma de uma tabela.

Base de Dados:

Será utilizada uma instância da Microsoft Azure SQL Database para armazenar toda a *data* operacional:

- Aceder (e proliferar) dados de configuração.
- Dados de contexto e dicionários.
- Ficheiros de áudio e dados analíticos gerados
- Dados de Logging e das APIs externas

Camada de Lógica de Negócio (Business Logic Tier)

Esta camada compreende todos os elementos software que controlam a aplicação executando o seu processamento:

APIs RESTFull:

Conjunto de APIs RESTFull usadas na comunicação e transferência de dados através de webservices expostos.

Todos os submodelos da camada de apresentação e também da API de processamento fazem uso das APIs RESTFull para enviar e receber dados da camada de Dados.
APIs RESTFull mais relevantes:

- API de autenticação;
- APIs Externas para tradução – Speech to Text (Bing Speech API e Speaker recognition API);
- API para Database querying;
- API para o funcionamento de Logging de informação;

Evalyze.VoC Audio Files Processor API:

Todo o processamento dos ficheiros de áudio:

- Conversão para o formato WAV
- Silence trimming;
- Noise reduction.
- File splitting;
- Processamento Multithreaded Speech-To-Text;
- Sincronização com o VoC through Evalyze.VoC RESTFull API

Workflow

Camada de apresentação (Presentation Tier)

Esta componente será responsável pela apresentação da aplicação.
A componente front-end da aplicação irá demonstrar todos os índices relevantes do processamento global, bem como gráficos que representam a performance da aplicação.

Componente de Auditoria (AuditBOT)

Esta componente reger-se-á bastante (quase integralmente) pelas funcionalidades desenvolvidas no VOC, pelo que o seu principal desenvolvimento dará início apenas quando a componente VOC estiver 100% funcional. Não obstante, estão a ser desenvolvidas funções na API de processamento que darão suporte à maior parte das funcionalidades do AuditBOT, tais como: separação áudio com base em vozes distintas e captura de índices de informação relevante (períodos de silêncio, períodos de falas do interveniente técnico).