NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Aplicações Avançadas de Instrumentação Biomédica

Relatório de Projeto

Interface de Cálculo controlada pela Fala

Luísa CastelBranco Silva, 55695
Tiago Coelho, 55097
Mestrado Integrado em Engenharia Biomédica

Dezembro 2022

Enquadramento e Contexto

O reconhecimento da fala refere-se à capacidade do computador de identificar palavras que foram ditas e de as converter em texto legível. Nos dias de hoje, *softwares* de reconhecimento de fala são largamente utilizados à escala mundial. O seu principal benefício está associado ao aumento de produtividade e prestabilidade, e à facilidade de uso. Cada vez mais, diversas tecnologias têm em si incorporadas esta capacidade de interpretar o que o utilizador diz e de atuar em conformidade.

Esta relevância do reconhecimento da fala, aliada a uma curiosidade de descobrir os métodos pelos quais esta é conseguida, levaram-nos à ideia de construir uma interface de cálculo controlada pela fala.

Programa e Funcionalidade

A base deste programa a implementar seria, então, um *website* a funcionar como interface gráfica para a calculadora, onde o utilizador seria capaz de introduzir, através da fala, uma expressão matemática, recorrendo a algoritmos de *Machine Learning*. Esta expressão poderia ser constituída pelos algarismos numéricos de 0 a 9, pelas operações matemáticas básicas (adição, subtração, multiplicação e divisão), bem como algumas outras mais complexas (potência e raiz quadrada). Também a conjugação de diferentes operações estaria ao alcance, através da introdução de parênteses. Ademais, procurando alargar as funcionalidades e, consequentemente, a complexidade do desafio, implementaram-se também as opções de expansão binomial e de representação gráfica (na qual as constantes da função são introduzidas por voz).

Idealmente, o utilizador conseguiria ditar, na íntegra, a expressão que pretende calcular. Contudo, isto consistiria num problema de classificação multi-classe com 18 classes (algarismos, 10, operações básicas, 4, e complexas, 2, e parênteses, 2). Apesar de ter sido implementado, a performance do classificador encontrava-se algo aquém do desejado e, portanto, a dimensão do problema foi reduzida para a identificação de, apenas, os 10 algarismos numéricos (sendo os restantes componentes introduzidos por botões da interface).

Dados de Treino

O *dataset* de treino foi obtido pela gravação de 20 áudios para cada classe, convertidos, posteriormente, em ficheiros de texto. Consiste, então, em 18 ficheiros *.txt*, onde cada ficheiro tem a si associada uma classe. Cada linha representa um áudio de 2 segundos, gravado a taxa de aquisição de 44100 Hz, pelo que cada ficheiro possui 20 linhas, cada uma com 88200 pontos. A voz presente nos áudios é sempre a mesma, no sentido de aumentar a precisão dos resultados. Tendo em conta o contexto pedagógico, aceitou-se este procedimento, apesar de um *overfitting* do modelo a uma voz específica estar a si inerente. No contexto "industrial", esta não seria uma solução adequada, no entanto, a alternativa seria obter um *dataset* bastante rico em frequências (timbre) e pronúncias, de modo a ser possível interpretar a fala de qualquer utilizador, o que não estava ao nosso alcance.

Extração e Seleção de Features

O ficheiros de texto onde os áudios se encontravam codificados foram convergidos num único ficheiro, e, posteriormente, num vetor numérico de dimensão 360, (18 classes $\times$ 20 repetições) em que cada posição representa um *array* de dimensão 88200 correspondente a um áudio. Foi criado um *array* 1D de dimensão 360 com as classes do primeiro respetivamente ordenadas.

Foi então realizado uma divisão entre conjuntos de treino e de teste organizando as amostras aleatoriamente e com uma proporção 70/30 respetivamente.

Biblioteca tsfel

Foram extraídas, numa primeira abordagem, 60 *features*, com recurso à biblioteca *Time Series Feature Extraction Library* (*tsfel*), sem qualquer tipo de seleção.

Orange

Depois de uma primeira filtragem para eliminar as *features* correlacionadas em mais de 95%, utilizou-se o *software Orange* para categorizar as melhores *features* individualmente, bem como os melhores modelos ou combinações de modelos classificadores.

Os modelos com melhor desempenho foram o *Random Forest* e um *Stacking Classifier* constituído pelo *Random Forest* anterior, um classificador *Naive Bayes*, e uma Rede Neuronal, para a maior parte das combinações de *features* testadas e parâmetros de classificadores utilizados. Deste modo, decidimos apostar um pouco mais nestes classificadores.

Exhaustive Feature Selector

Como os melhores 5 desempenhos individuais não têm de resultar na melhor combinação de 5 *features*, foi realizada também uma abordagem de *exhaustive feature selector* de modo a obter a melhor combinação com um máximo de 15 *features* para o classificador de *Random Forest* e de *Stacking* com base na *accuracy* resultante. De modo a acelerar o processo da *exaustive feature selector*, procuraram-se as combinações dentro das 15 *features* que mostravam melhor desempenho individual no *Orange*.

Apesar de o *exhaustive feature selector* ser preferencial quando tempo não é um problema, restringir o modelo às 15 melhores limita o seu desempenho. Usou-se, então, mais uma abordagem para ser possível iterar sobre algumas combinações de *features* mas utilizando todas. O *forward feature selector* utiliza um algoritmo de seleção iterativo de modo de modo a favorecer, em cada iteração, a melhor combinação de N *features*, recombinando a com todas as restantes na seguinte iteração ficando com um tamanho de $N + 1$. O limite imposto foram 20 *features*, no entanto, preferencialmente, o algoritmo termina quando o desempenho da melhor combinação de N *features* é inferior ao da melhor de $N - 1$ anterior.

Classificador

De um modo geral, o classificador *Random Forest* resultou melhor para o nosso *dataset*, sendo que o processo de seleção com melhor desempenho foi o FFS. Foram utilizadas um total de 12 *features*.

Depois de fixas, as *features*, recorreu-se a um *grid search* de modo a obter os melhores parâmetros para o modelo em questão.

Resultados

A *accuracy* do *training set* para o *Random Forest* no *exhaustive feature selector* limitado a 15 features foi de 0.85, para o FFS sem limite foi de 0.96, e para o mesmo FFS após *grid search* foi 0.98. No entanto, a *accuracy* final para test set foi de 92.2%, o que é bastante bom, dado que é um problema multi-classe com 18 possíveis *outputs*. O classificador aleatório acertaria 10% das tentativas.

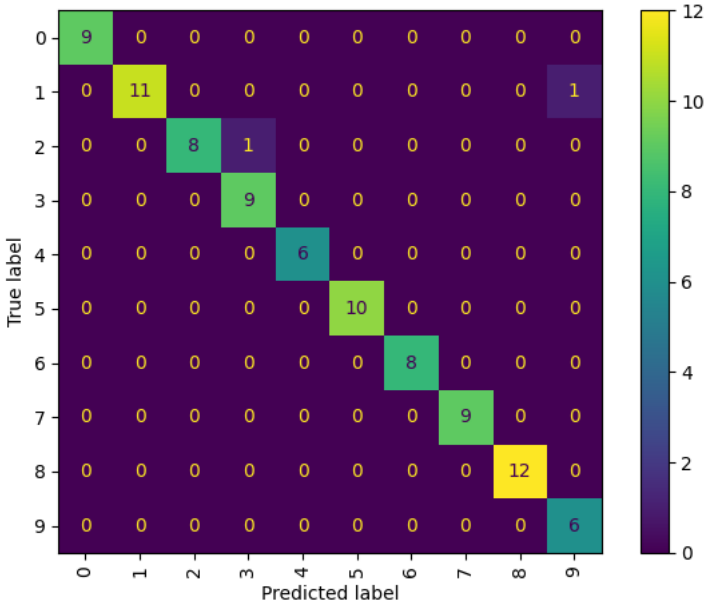

Figura 1: Matriz de confusão do classificador.

Na Figura 1 está a matriz de confusão do classificador. Conclui-se que este demonstra uma *performance* mais do que razoável, uma vez que a matriz de confusão é praticamente diagonal. Isto significa que, na maioria dos casos, a classificação pelo modelo e a classe real coincidiram, ou seja, que o modelo acertou, na maioria das vezes.

Funcionamento

O classificador foi implementado num ficheiro *.py*, recorrendo à biblioteca *scikit-learn*. O modelo treinado foi depois exportado para um ficheiro *.joblib*, através da biblioteca *joblib*. Um outro *script Python* importa este modelo, recebe o comando que inicia a gravação de um áudio de 2 segundos, classifica-o, e envia a correspondente classificação. Esta comunicação é estabelecida pelo protocolo *MQTT* e as bibliotecas associadas. A gravação de áudio é conseguida através da bibloteca *sounddevice*.

A interface em si é uma página *streamlit* que integra componentes de *Python* e de *html*. Esta constitui a interação direta com o utilizador, que pressiona o botão **REC** para gravar o algarismo, e os associados às operações que pretende executar. Por detrás desta página, está um *script* auxiliar, responsável por receber a classificação proveniente do modelo de *Machine Learning*.

A implementação no *Raspberry Pi* (RPI) consiste em utilizar este pequeno computador para correr os *scripts* centrais do projeto. Em particular, o RPI corre o código responsável por receber o comando *start* que inicia a gravação do áudio, que o classifica, e que envia a classificação. Para além disto, é também onde se abre a página *streamlit* diretamente.

O *streamlit* é uma plataforma que permite o desenvolvimento simplificado de aplicações *web*, dirigida, especialmente, para manuseamento de dados. Apresenta vários componentes de fácil implementação e permite, também, elementos de *html* e *JavaScript*.

No início da página *streamlit* desenhada para este projeto, encontra-se o módulo da calculadora.

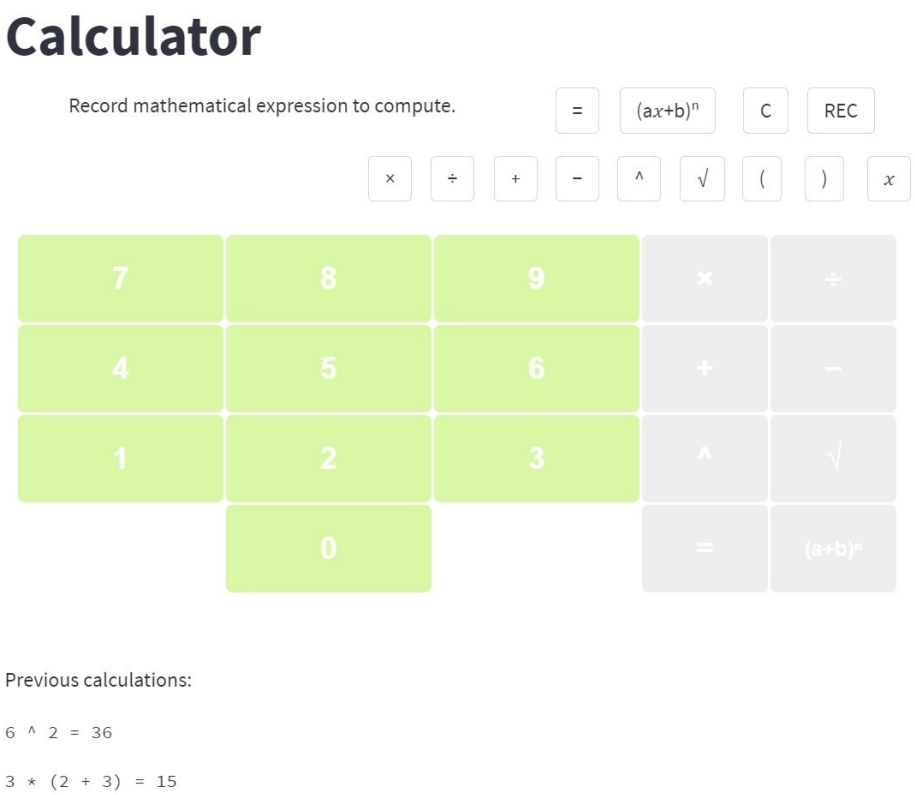

Figura 2: Parte da interface gráfica associada à calculadora.

Quando o utilizador pressiona o botão **REC**, o comando *start* é enviado e a gravação começa. O resultado da classificação, ao ser recebido, é gravado em formato *string* num ficheiro *.txt*. A página, ao fim de algum tempo, é atualizada, lê este ficheiro, e mostra o resultado da classificação no mostrador da calculadora. De seguida, o utilizador pressiona o botão da operação que pretende realizar, e a *string* associada é adicionada ao ficheiro *.txt*, em frente ao algarismo já presente. O seguinte algarismo é gravado do mesmo modo. Por fim, prime-se o botão do sinal de igual, que é responsável por calcular e mostrar o resultado da operação. Para tal, recorre à função *eval()* do *Python*, que recebe a expressão em *string* e a interpreta como numérica. Para além disto, a expressão calculada e o seu resultado são guardados num outro ficheiro, para que seja possível visualizar os cálculos realizados anteriormente. Qualquer expressão inválida resulta num erro de sintaxe.

Para se tirar partido da expansão binomial, introduz-se uma expressão do tipo $(ax + b)^n$ e, ao invés de o utilizador pressionar o sinal de igual, pressiona o botão que exibe esta expressão. O resultado é obtido através da função *expansion()*, que recebe a expressão também em formato *string* e a calcula. Esta função, por nós implementada, advém do *kata* "Binomial Expansion" do *CodeWars*, que se considerou relevante incorporar no projeto.

O botão **C** permite ao utilizador limpar o mostrador da calculadora, caso queira recomeçar a expressão. Concretamente, este comando elimina a informação presente no ficheiro onde são escritas as expressões, retornando ao estado inicial *default*.

Abaixo dos botões interativos está um elemento *html* similar a uma calculadora. Idealmente, os botões seriam funcionais mas, para tal, é necessário estabelecer a comunicação bidirecional *JavaScript/Streamlit*, o que não foi possível por limitação de tempo.

O módulo que procede a calculadora corresponde à funcionalidade de representação gráfica.

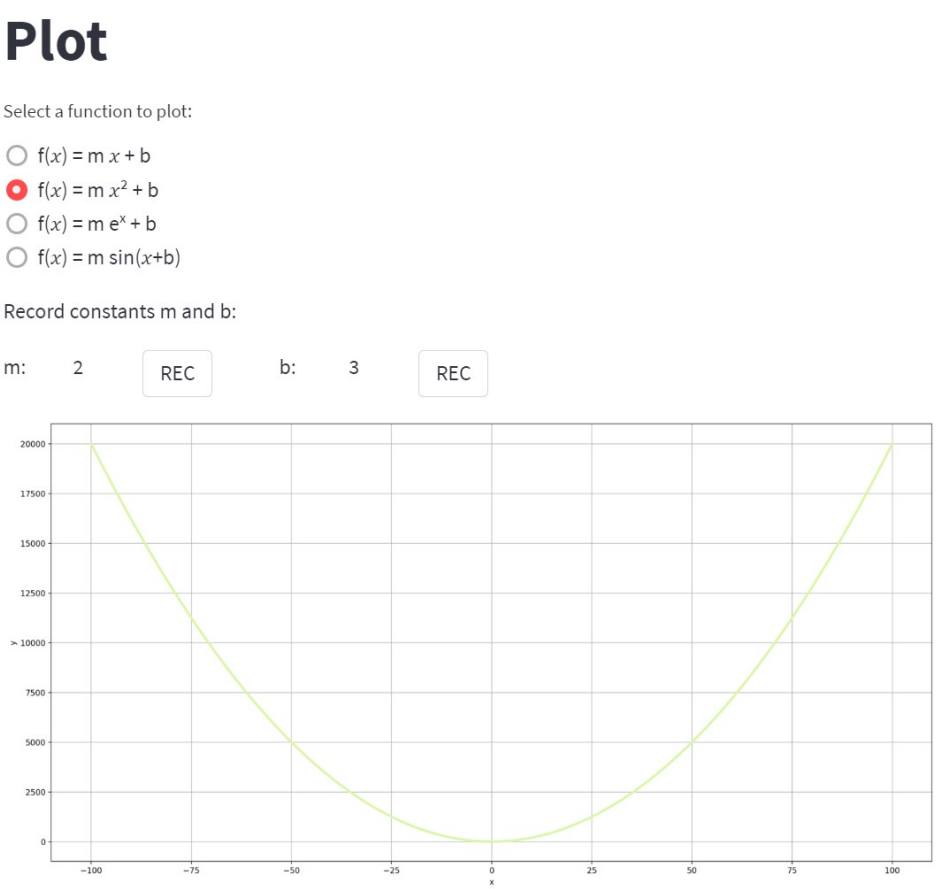

Figura 3: Parte da interface gráfica associada à representação.

Aqui, são propostas ao utilizador quatro funções matemáticas conhecidas: as funções afim, quadrática, exponencial e trigonométrica (em particular, a função seno). O utilizador pode, de seguida, escolher as constantes associadas, através de botões **REC**, do mesmo modo que anteriormente. O gráfico associado à função escolhida e com os parâmetros obtidos por reconhecimento de fala é um elemento de *Python*, conseguido com recurso à biblioteca *matplotlib*.

Conclusão

O balanço geral do projeto foi bastante positivo, tendo sido possível treinar um classificador de *Machine Learning* e obter um modelo com uma *performance* bastante satisfatória. Para além disto, foi implementada, com sucesso, a interface de cálculo no *streamlit*, com várias funcionalidades diferentes.

Para conseguir este resultado final, conhecimentos de várias áreas foram integrados e várias ferramentas utilizadas. Em primeiro lugar, a capacidade de programação em *Python* foi fulcral em quase todos os passos. Os conceitos presentes de *Machine Learning* foram traduzidos em código, tendo-se recorrido ao *software Orange*, e a biblitecas como a *tsfel*. Muitas outras competências foram também desenvolvidas, como a capacidade de lidar com ficheiros e o estabelecimento de protocolos de comunicação (em concreto, *MQTT*). Também algum conhecimento de *html* foi posto em prática, para a apresentação do *website*.