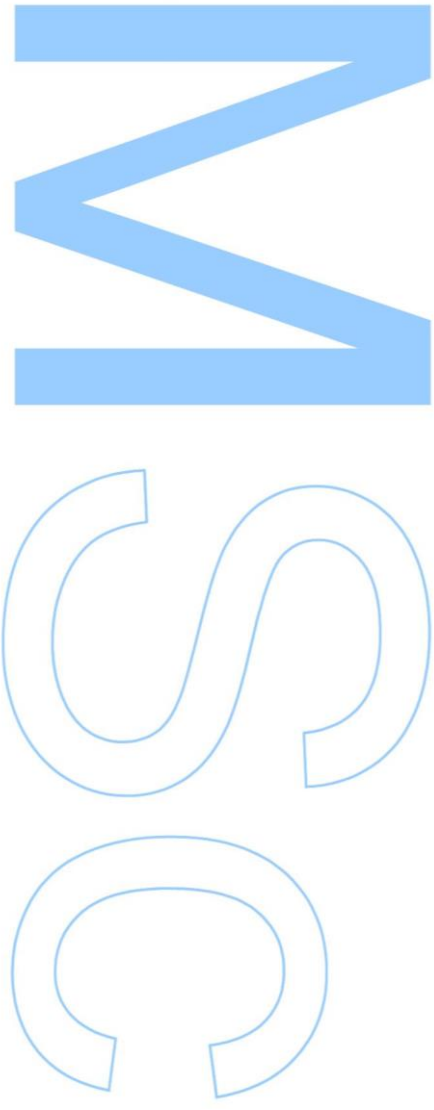
Integração e Processamento de Dados em Experiências sobre Processos Cognitivos de Escrita

Patrícia Sofia Borges Santos
Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos
Departamento de Ciências de Computadores
2016

Orientador
José Paulo de Vilhena Geraldes Leal, Professor Auxiliar, Faculdade de Ciências da Universidade do Porto



U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

À minha avó que sempre esteve ao meu lado

Agradecimentos

Quero agradecer aos meus pais por sempre me apoiarem nas várias etapas da minha vida e por estarem sempre presentes.

Ao meu namorado Jorge por estar sempre presente e me ajudar mesmo nos momentos menos bons.

Aos meus amigos Tânia, Joana e Nuno que sempre me proporcionaram momentos de diversão e me ajudaram a recarregar energias.

Aos meus colegas e professores que tive o prazer de conhecer durante o mestrado e que me ajudaram a crescer como profissional e como pessoa.

Ao meu orientador José Paulo Leal pela paciência e ajuda durante a elaboração da tese e também a todos os membros do projeto DAAR com quem tive o prazer de trabalhar.

A todos os que passaram pela minha vida nesta fase e de alguma forma me marcaram e fizeram crescer.

Resumo

A escrita existe há milhares de anos e tem evoluído ao longo do tempo. Mesmo assim, continua a ser objeto de estudo e analisada para os mais variados fins. A importância da escrita é grande podendo influenciar o nosso sucesso académico e até profissional. Com o intuito de ajudar pais e professores no ensino da escrita foi criado o projeto Desenvolver, Automatizar e Autorregular os Processos Cognitivos na Composição Escrita (DAAR). Este projeto de investigação científica é financiado pela Fundação para a Ciência e a Tecnologia (FCT). Para dar suporte a esta investigação foi criada a plataforma HandSpy que permite aos investigadores analisar os dados recolhidos através de *smartpens*. Esta plataforma permite gerir todas as fases de uma experiência, desde a criação de tarefas até à geração de resultados.

Com esta tese pretende-se apresentar os problemas da escrita manual e que tipo de informação se poderá extrair de um texto manuscrito recolhido com o equipamento utilizado para a plataforma HandSpy. Esta informação poderá facilitar o trabalho de análise por parte dos investigadores fornecendo-lhes mais dados de forma automática. Este trabalho será feito fora da plataforma mas utilizando a mesma estrutura de dados desta de forma a posteriormente se integrarem as novas funcionalidades no HandSpy para uso dos investigadores.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

A escrita é algo essencial na vida de uma pessoa. Desde o sucesso académico e profissional ao pessoal, saber escrever é uma competência essencial em que cada pessoa tem o seu ritmo de aprendizagem e algumas têm mais dificuldades que outras. Desta forma, foi criado o projeto Desenvolver, Automatizar e Autorregular os Processos Cognitivos na Composição Escrita (DAAR) que sabendo o grande desafio que é o ensino da escrita, acredita que este pode ser superado com a colaboração entre professores, alunos, pais e investigadores e também com a criação de práticas pedagógicas assentes em evidências científicas. Dentro deste projeto existe a plataforma *online* HandSpy[7, 8], que surgiu como uma ferramenta para o processamento e análise de textos recolhidos pelos investigadores.

1.1 Contextualização

1.1.1 Outras plataformas

Esta secção pretende dar a conhecer duas plataformas de desenvolvimento colaborativo para gestão de experiências em processos cognitivos na escrita. O estudo de processos cognitivos tem sido suportado por ferramentas de recolha digital da escrita e posteriormente outras ferramentas para a análise destes dados. Visto ser este o foco do HandSpy, foram estudadas ferramentas de análise antes e durante a criação da plataforma.

A plataforma Eye and Pen[1] teve originalmente o objetivo de estudar a leitura durante

a produção de texto. Ou seja, a recolha de dados foca-se não apenas no material de escrita como também na recolha dos movimentos oculares do sujeito para que possam depois ser analisados em conjunto de forma sincronizada. Isto permite aos investigadores não só analisar o controlo visual em relação à escrita como também revisões feitas pelo utilizador e até a consulta de outras fontes. Para a recolha destes dados é necessário um *hardware* que registe os movimentos oculares e outro que registe os movimentos da caneta. Este tipo de equipamentos está dependente das especificações de quem os produz e da sua utilização cuidada o que causa algumas dificuldades. Por exemplo, nalguns casos a recolha dos movimentos oculares obriga a que o sujeito esteja com a cabeça na mesma posição para que os dados sejam recolhido corretamente. Além disso a utilização de vários aparelhos pode distrair e interferir no processo de escrita podendo assim invalidar o estudo.

Outra plataforma estudada foi o Ductus[5]. A plataforma é constituída por uma mesa digitalizadora que disponibiliza os dados sobre a escrita *online* para análise. A ferramenta *online* está dividida em dois módulos independentes. O primeiro módulo é particularmente adequado a crianças e pacientes com patologias na escrita pois foca-se na apresentação de estímulos. Já o outro módulo permite analisar dados como a velocidade, duração, fluência, pausas e a trajetória da escrita.

1.1.2 HandSpy

A plataforma *online* HandSpy analisa dados recolhidos no formato Ink Markup Language (InkML)[11] que são obtidos através de canetas digitais da marca *Livescribe*[1] como a da figura 1.1. Estas canetas funcionam com um formato de papel especial que utiliza x e y como pontos de referência obtendo assim a posição exata de forma análoga a um *gps*. A informação retirada contém os traços feitos e os respetivos *timestamp* de cada ponto.

Para a recolha de dados além das canetas digitais são necessárias duas aplicações de *desktop*. Uma para a instalação do programa de recolha nas canetas e a outra para extrair os ficheiros gerados. Este programa instalado nas canetas, denominado de *penlet*, permite a utilização de formatos de folhas personalizados para os investigadores. Além disso, também é possível recolher informação como os tempos de cada ponto que são importantes para o processamento de alguns dos dados pedidos pelos investigadores. A estrutura das folhas especiais é definida através de um ficheiro

[1] https://www.livescribe.com/pt/smartpen/echo/

Figura 1.1: Livescribe Smartpen

ArisFlow Diagram (AFD) que permite descrever o formato da folha e definir zonas especiais, para posterior programação da caneta. No final do processo de recolha é exportado um ficheiro no formato *InkML* que contém o conjunto de vários *strokes* 1.2. Um *stroke* é o conjunto de pontos recolhidos de forma continua até que a caneta seja levantada. Desta forma, cada *stroke* tem um conjunto de pontos com os respetivos *timestamps*.

```
<trace>
 2534 685 37297520816, 2537 684 37297520829, 2539 681 37297520843,
 2544 678 37297520856, 2546 677 37297520883, 2546 678 37297520896,
 2548 680 37297520909, 2549 682 37297520923, 2554 695 37297520949,
 2559 707 37297520963, 2565 720 37297520976, 2572 732 37297520989,
 2580 749 37297521016, 2582 753 37297521029, 2584 754 37297521043,
 2584 754 37297521056, 2584 754 37297521083, 2584 754 37297521096,
 2584 754 37297521109, 2584 754 37297521123, 2584 753 37297521149,
 2585 751 37297521163, 2586 748 37297521176, 2592 738 37297521189,
 2603 710 37297521216, 2611 694 37297521229, 2615 684 37297521243,
 2620 673 37297521256, 2620 670 37297521283, 2621 670 37297521296
</trace>
```

Figura 1.2: Exemplo de um *stroke* de um ficheiro InkML

Após a recolha, é utilizada a plataforma *online* do HandSpy para ser feita uma análise dos dados. De forma a facilitar a sua utilização, a plataforma foi desenvolvida com os seguintes conceitos:

Protocolo: O Protocolo é constituído pelo conjunto de *strokes* e é único para cada participante de uma determinada tarefa.

Tarefa: Os vários protocolos podem ser separados em várias tarefas que podem ser, por exemplo, narrativas, ditados e cópias.

Participantes: Os participantes produzem o texto que é o alvo da investigação.

***Burst*:** O *burst* é o tempo que o participante escreve continuamente até existir uma pausa.

Pausa A pausa é o intervalo de tempo em que o participante terminou de escrever até que recomece.

Threshold: Este parâmetro permite definir o conceito de pausa, que por definição é de 2 segundos. Desta forma podem ser feitos vários tipos de testes com pausas diferentes.

Na aplicação do HandSpy, o investigador tem duas fases de trabalho: carregamento e análise. Na primeira é necessário realizar o carregamento dos ficheiros InkML retirados das canetas para a plataforma *online*. Este passo é feito através da *tab Upload* mostrada na figura 1.3. Após o carregamento, o utilizador terá de atribuir a cada protocolo a respetiva tarefa e um código que o identificará daí em diante. Finalmente, pode começar a análise dos textos recolhidos. Nesta análise é apresentada uma tabela com os dados do protocolo a analisar e uma imagem com o texto 1.4. Na imagem é possível identificar o início e fim dos *bursts* assim como das pausas. O investigador pode editar os dados e alterar os parâmetros de forma a ajustar a informação de acordo com a sua investigação e no final descarregar um ficheiro *excel* com os dados processados.

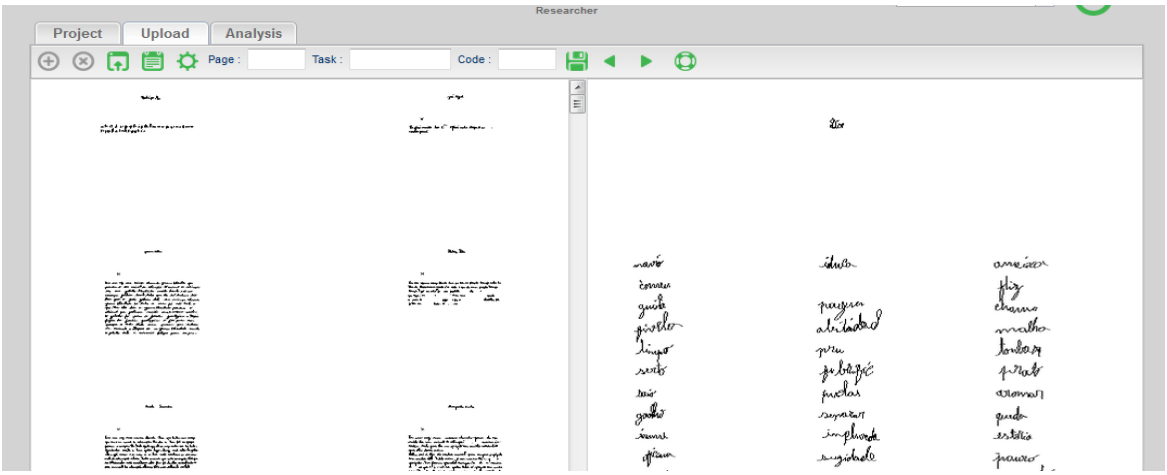


Figura 1.3: *Tab Upload*

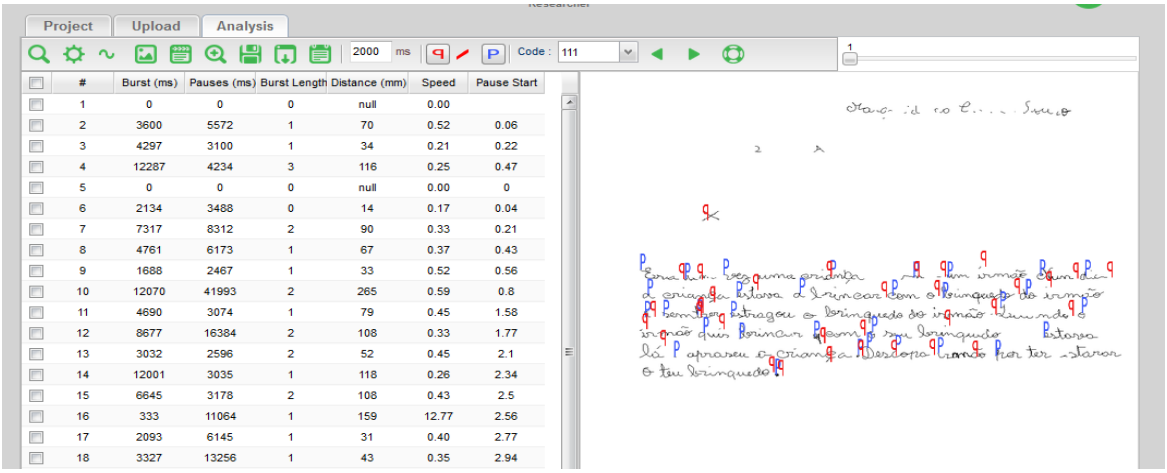


Figura 1.4: *Tab* de análise do HandSpy

Também é possível ao investigador definir informações básicas do seu projeto, criar tarefas e adicionar participantes. Estas opções estão disponíveis na *tab* inicial *Project* como ilustra a figura 1.5.

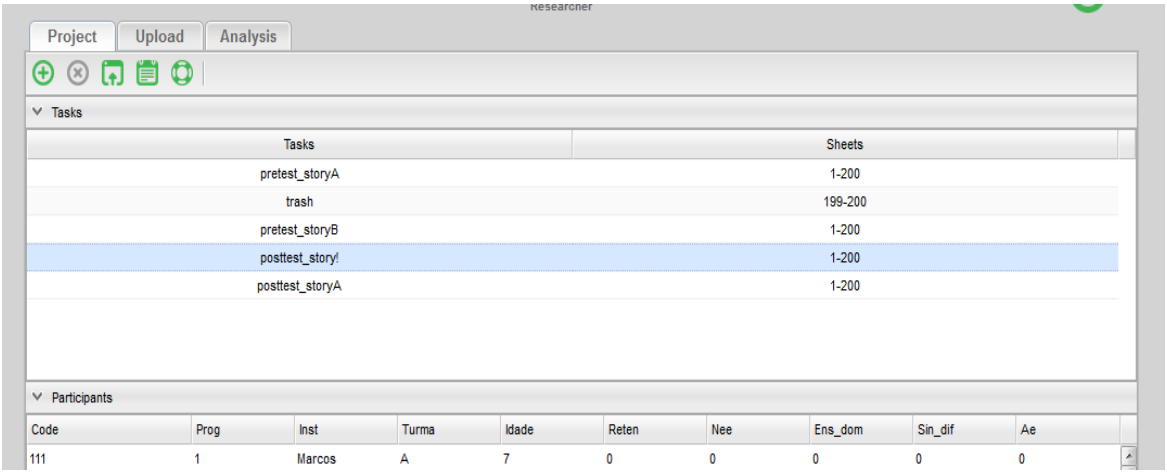


Figura 1.5: *Tab Project*

Estas são as ferramentas usadas atualmente pelos investigadores do projeto DAAR e também outros investigadores de vários países que aderiram a esta plataforma *online*. É, por isso, importante que esta aplicação esteja em continuo melhoramento e seja importante o desenvolvimento de novas funcionalidades de acordo com as opiniões e pedidos de investigadores.

1.2 Motivação

No HandSpy são realizadas várias experiências com os mais variados tipos de pessoas, desde adultos a crianças. No caso dos investigadores do projeto DAAR, as experiências são focadas nas crianças e na análise da sua escrita de forma a criar programas que promovam o seu ensino. Além disso, esta plataforma é também utilizada por outros investigadores para os mais variados estudos, como uma única ferramenta ou até em conjunto com outras existentes nos projetos. Foi desta forma verificado que seria interessante a plataforma deixar de ver os textos como simples traços e começar a extrair informação como linhas, palavras e até o reconhecimento de letras. Esta funcionalidade irá permitir aos investigadores o acesso a dados considerados relevantes de forma automática, retirando-lhes a necessidade de o fazerem manualmente.

1.3 Objetivos

O objetivo principal deste trabalho é implementar um algoritmo para o reconhecimento da escrita manual. O objetivo final deste algoritmo será o reconhecimento de letras e a partir daí de palavras. No entanto, como este objetivo poderá não ser alcançável, foram definidas etapas. Inicialmente, será preciso dividir o texto em linhas e depois em palavras (sem reconhecer qual a palavra escrita). Chegando a este ponto, já será possível fornecer dados como o número de palavras por *burst*. O objetivo final será a tentativa de reconhecimento de letras e posteriormente as palavras, tentando alcançar o melhor resultado possível. Outro objetivo proposto foi o melhoramento da aplicação HandSpy, fornecendo suporte e desenvolvendo novas funcionalidades.

1.4 Organização

Esta tese está organizada da seguinte forma. Inicialmente será apresentado o estado da arte no reconhecimento de escrita manual. De seguida é abordada a metodologia e implementação do reconhecimento de escrita assim como os testes feitos e a sua interpretação. Finalmente é apresentada a conclusão da tese sobre o trabalho feito ao longo desta e os trabalhos futuros que poderão resultar deste projeto.

Capítulo 2

Estado da arte

Este capítulo aborda os problemas e limitações do reconhecimento de escrita manual assim como abordagens ao problema.

2.1 Limitações

O reconhecimento de escrita manual é um tema já muito estudado e que representa um grande desafio pela sua complexidade. A escrita existe há milhares de anos e tem evoluído ao longo do tempo. Na escrita manual podem existir dois estilos: cursiva ou impressa. Na escrita impressa, a pessoa escreve cada letra separadamente e com o estilo de letra semelhante ao utilizado, por exemplo, na prensa móvel. Já o estilo cursivo é projetado para a agilidade na escrita e no caso do alfabeto latino as letras são interligadas para formar palavras.

Adicionalmente, a forma como se escreve não só depende do tipo de alfabeto e cultura em que se vive como também da personalidade, idade, estado emocional e até a mão com que a pessoa escreve. Um exemplo disto é a forma como um 1 e o um 7 podem ser confundidos quando escritos por pessoas e culturas diferentes. Cada indivíduo tem uma forma única de escrever, razão pela qual é pedida a assinatura como comprovativo em muitos casos (cheques, contratos, etc). Isto faz com que seja necessário ter em conta todas as variantes que uma simples letra pode ter. Tornando muito complexa a tentativa de reconhecer uma letra corretamente, pois por maior que seja a base de dados de letras, existe o problema de uma letra parecer outra completamente diferente. Até os humanos, por vezes, têm de ler o texto inteiro de outra pessoa para perceber,

através do contexto, algumas das letras escritas.

A forma como se escreve também dificulta a separação de palavras em letras. Isto acontece pois, por vezes, temos tendência a escrever com uma certa inclinação, ficando assim algumas letras sobrepostas a outras. Aumentando a dificuldade de detetar o inicio e fim de uma letra, principalmente em casos onde não é possível diferenciar de forma temporal os traços.

2.2 Abordagens

Existem dois tipos de reconhecimento de escrita: *offline* e *online*[10]. O primeiro baseia-se em texto digitalizado como cheques e documentos assinados. Já o *online* consiste na tentativa de reconhecimento da escrita à medida que o utilizador vai escrevendo, obtendo-se desta forma uma noção de temporalidade como acontece, por exemplo, na utilização de canetas digitais. Os dados escritos com uma caneta digital são guardados com informação como a posição e tempo, que poderão depois ser utilizados para algoritmos de reconhecimento. Por esta razão, serão focadas as abordagens *online*. As técnicas pioneiras nesta área utilizavam redução de hipóteses através de formulários específicos. Por exemplo, num campo onde se terá de escrever o código postal, podemos reduzir as hipóteses de caracteres a números (0-9) sendo assim mais fácil o seu reconhecimento. No entanto, estas técnicas não são adequadas para o problema deste trabalho visto que o objetivo é reconhecer qualquer tipo de texto que uma pessoa escreva, o que impossibilita uma redução de hipóteses.

No reconhecimento de caracteres existem duas formas de o fazer: Optical Character Recognition (OCR)[3] e Intelligent Character Recognition (ICR). OCR consiste no reconhecimento ótico de caracteres a partir de um arquivo de imagem ou mapa de bits. Dependendo do algoritmo utilizado este tipo pode ter várias funcionalidades tais como:

- Reconhecimento automático de matrículas
- Criar versões digitais (em texto) de documentos impressos
- Reconhecimento de textos manuscritos

Para casos mais complexos existe o ICR. Este sistema permite que diferentes estilos de escrita sejam aprendidos durante o processamento de forma a melhorar os resultados

obtidos. A maioria dos ICR usam um sistema baseado em redes neuronais, que é automaticamente atualizado com novos padrões de escrita. Normalmente é utilizado para formulários impressos ou desenhados através de ferramentas de desenho. Para a identificação de caracteres utilizam-se técnicas de reconhecimento de padrões tais como: estatísticas (teoria de decisão), sintéticas (estruturais) e baseadas em redes neuronais.

A compreensão de como a escrita manual é gerada é importante tanto para o desenvolvimento de métodos *online* como *offline*. Vários modelos foram propostos ao longo dos anos, podendo estes ser divididos em duas classes gerais: top-down e bottom-up. Modelos *top-down* focam-se em processar informação de alto nível como a semântica. Já na abordagem bottom-up o interesse está na análise e sintetização de processos de baixo nível envolvidos na produção de um *stroke* e depois começando a subir gerando grafos, palavras, etc.

De forma generalizada, o processo de reconhecimento de escrita manual tem as seguintes etapas como se pode ver na figura2.1: pré-processamento, segmentação, extração de *features*, classificação, rotulação, pesquisa no modelo de linguagem e pós-processamento do modelo de linguagem [2].

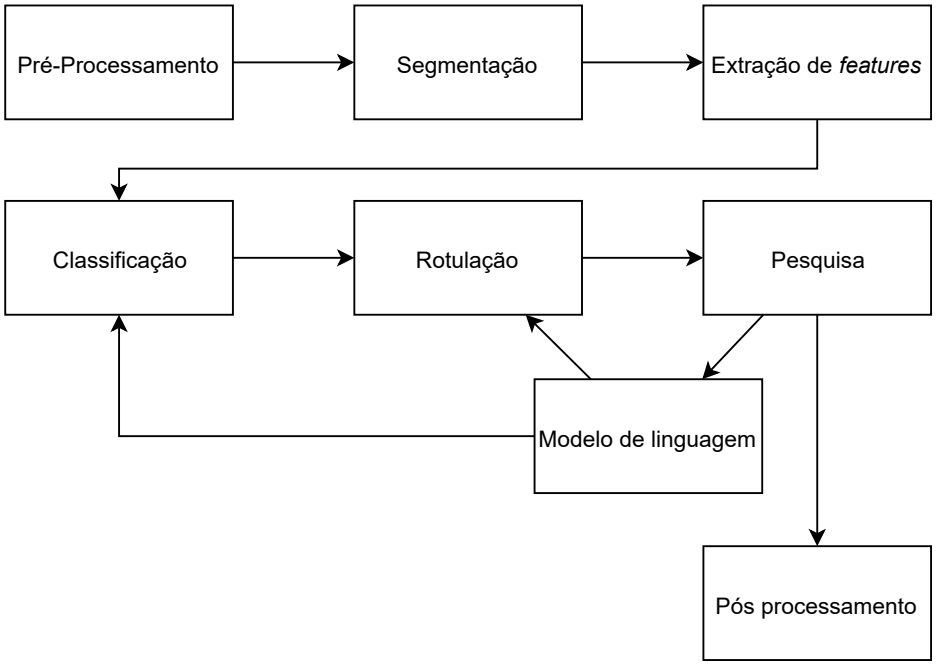


Figura 2.1: Processo genérico do reconhecimento de escrita manual

Pré-processamento

Alguns *tablets* que fazem a digitalização da escrita aplicam filtros ao nível do *hardware* aos dados recolhidos. Estes filtros podem por vezes impedir que o reconhecimento de letras ocorra corretamente pois "limpam"detalhes que tornam as letras únicas[2]. É por isso aconselhável deixar a filtragem dos dados para o algoritmo de reconhecimento.

Depois desta filtragem inicial, é preciso definir uma altura padrão de forma a que o esquema de reconhecimento seja independente do tamanho[2]. Para isso, é necessário definir a linha de base e a linha intermédia como se pode ver na figura 2.2. A área resultante destas duas linhas define o espaço onde terá de existir sempre traços e por isso mesmo torna esta zona mais confiável para a normalização de tamanho.

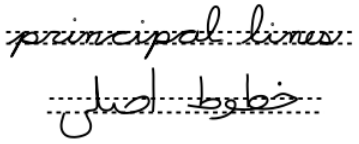


Figura 2.2: Exemplo da definição da linha base e intermédia [2]

Outras possíveis normalizações são a correção da inclinação e do declive. Para a correção da inclinação é preciso calcular a média do declive com uma orientação vertical. Os dados do declive são depois compensados usando a diferença entre o valor computado e o declive do eixo vertical. Esta correção é feita através do corte visto que em casos de pequenos defeitos o corte fornece uma boa aproximação para a rotação. A correção do declive é um processo iterativo que utiliza as normalizações anteriores para estimar o declive da linha base até que esta seja horizontal. Também podem ser aplicadas técnicas para remover pontos duplicados, ou seja, pontos que têm as mesmas coordenadas assim como *hooks*[4]. *Hooks* é o termo usado para um conjunto de pontos dentro de um *stroke* que são apenas ruído e que normalmente encontram-se no início e fim de um *stroke*. Este ruído pode acontecer por diversas razões como imprecisões na deteção da caneta no dispositivo ou o movimento rápido e errático de quem escreve no dispositivo[6].

Segmentação

O input da escrita manual pode variar dependendo das situações. Na figura 2.3 temos como exemplo 5 tipos de escrita. No primeiro exemplo, *boxed discrete*, o utilizador escreve cada letra no retângulo respetivo já existente. Este poderá ser a

forma mais simples de segmentação de caracteres. Sem a utilização de caixas para separar os caracteres mas ainda uma forma simples de as separar será o exemplo seguinte, *discrete*, em que o utilizador escreve cada letra separada uma da outra. Neste exemplo, para resolver o problema da segmentação é preciso encontrar o espaço em branco deixado entre cada caracter, que poderá ser pré-definido ou obtido através da estatística do *threshold*. No caso do textitrun-on, a separação de caracteres torna-se mais complicada pois já não é possível fazê-lo através dos espaços entre letras pois podem não existir. É assumido, no entanto, que a caneta é levantada sempre que um caracter é escrito. Uma abordagem neste caso é tratar o *stroke* como a unidade mais pequena da palavra (letra) e conduzir uma pesquisa pelas combinações possíveis das etiquetas de *strokes* de forma a verificar se é possível criar uma palavra correta. No exemplo seguinte, *cursive*, são impostas apenas duas regras: que a caneta seja levantada no fim de uma palavra e todos os caracteres estão ligados entre si. Mas o maior desafio é a mistura dos dois últimos exemplos que é o caso para a maioria das pessoas. Neste caso apenas podemos assumir que eventualmente a caneta poderá ser levantada no final de uma palavra.

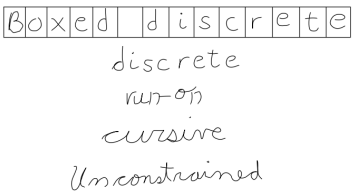


Figura 2.3: Exemplos de tipos de *input* na escrita manual [2]

Extração de *features* e classificação

Depois de segmentado, é realizada a extração de *features* das unidades, que é essencial para o algoritmo de classificação. Estes segmentos normalmente têm informações como as coordenadas dos pontos e um *timestamp*, que permitem saber a ordem e a velocidade desses pontos. Algumas das *features* procuradas são as diferenças entre as coordenadas adjacentes, o ângulo da linha tangente e também a curvatura de cada ponto (que podem ser avaliadas recorrendo aos pontos adjacentes). A análise das coordenadas de acordo com a linha base calculada anteriormente pode ajudar a distinguir letras maiúsculas de minúsculas. A informação das coordenadas também permite identificar conjuntos de pontos em áreas específicas, sendo este tipo de informação normalmente utilizado em sistemas *offline*.

Rotulação

Depois de extraídas as *features* dos respetivos segmentos, é altura de rotulá-los. Cada *stroke* (que pode ser considerado um subconjunto de um caracter) tem associado uma lista de possíveis caracteres de acordo com as suas características. Estas listas são criadas com a ajuda de classificadores tais como: *template matching*, estatístico e redes neuronais.

No caso do modelo de *template matching*, numa fase inicial de treino, os sistemas criam certos *templates* de segmentos de escrita, representando-os com os parâmetros das *features* usadas para o reconhecimento. Já na fase de reconhecimento, o sistema calcula uma medida de distância entre os *templates* e os segmentos de forma a gerar um valor de aproximação entre o segmento e o *template*.

Modelos estatísticos como, por exemplo, hidden Markov models (HMM), que anteriormente eram utilizados no reconhecimento de discursos, estão agora a ser usados no reconhecimento de escrita. Neste caso, são criados modelos de Markov de vários caracteres durante a fase de treino.

Na rotulação também têm sido utilizadas redes neuronais. Os sistemas baseados em redes neuronais apenas conseguem ser aplicados em escrita cursiva quando contêm os *templates* de todas as palavras.

Pesquisa

Depois de criada a lista de possíveis caracteres é altura de procurar a escolha que tem mais probabilidades de estar correta. Muitos sistemas utilizam técnicas de programação dinâmica para encontrar a escolha mais correta. A maioria destes métodos acaba por verificar todas as possibilidades antes de selecionar a mais correta o que pode ser demorado. Uma técnica útil que tem sido utilizada é o algoritmo A*. No entanto, como esta técnica vai "escolhendo"as possibilidades com maior probabilidade de estarem corretas, podem acontecer casos em que um caminho é inicialmente menos promissor e no final seria o mais correto mas que foi eliminado nas primeiras rondas em detrimento de outro com maior probabilidade de sucesso.

Modelo de linguagem

Uma forma de reduzir o tempo de pesquisa é a utilização de um modelo preditivo. Este modelo define as probabilidades de um caracter se seguir a outro de forma a eliminar caminhos que provavelmente não poderão estar corretos. Estas probabilidades vêm das estatísticas da linguagem em questão, por exemplo, no caso da língua portuguesa a letra "q"tem uma grande probabilidade de ser seguida pela letra "u". Este tipo de estatísticas permitem diminuir o número de caminhos a ser analisados pelos algoritmos de pesquisa.

Outro tipo de modelo é o de *templates*. Este modelo é utilizado em casos em que o texto está dentro de um grupo restrito de possibilidades, como por exemplo, o campo onde o utilizador deve escrever a matrícula de um carro. Normalmente, cada país tem um formato especifico para este tipo de dados o que diminui as possibilidades ao método de pesquisa.

Pós-processamento

Este processamento permite retificar as palavras retornadas pelo método de reconhecimento. Para isso, é utilizado um dicionário das palavras mais frequentes de forma a encontrar a string mais parecida com a retornada, aumentando assim a percentagem de acerto.

Curvas B-spline

No processo de escrita, todos os caracteres são formados por curvas de forma a que podemos considerar cada palavra e/ou letra como uma curva. Na modelagem geométrica, B-spline é uma das mais eficientes formas de representar curvas. O desafio reside em reconhecer o conjunto de pontos de controlo no caracter que definem a B-spline. Tendo um conjunto de pontos a formar as curvas dos caracteres, o primeiro passo (depois de feito o pré-processamento e segmentação) é a redução destes pontos de forma a que a nova curva seja quase igual à curva inicial. Um algoritmo que pode ser utilizado para a redução destes pontos é o Bandwidth[9]. Para fazer a correspondência de caracteres, começa-se por reconhecer e identificar caracteres de teste com os pré-definidos. É utilizada a distância euclidiana para avaliar a diferença entre dois pontos de controlo e assim comparar dois caracteres. Foi utilizado um conjunto de teste

constituído por algarismos escritos à mão, chegando a uma média de 95% de acerto [9]. Este método também obteve bons resultados no reconhecimento de assinaturas identificando, no entanto, a necessidade de existir um conjunto grande de dados para treino para que os resultados sejam positivos.

Capítulo 3

Reconhecimento de Escrita

A plataforma HandSpy, apesar de já processar dados sobre os textos recolhidos, não os vê como texto mas sim como um conjunto de traços. Neste capitulo será explicada a abordagem feita para que a plataforma passe a analisar os traços e a identificar neles características de um texto como linhas, palavras e posteriormente o reconhecimento de letras.

3.1 Metodologia

Antes de apresentar a metodologia, é importante explicar que tipos de dados são analisados e processados. Neste projeto, os dados referentes à escrita são guardados no formato digital InkML. Este é um formato *standard*, publicado em 2011 pelo *World Wide Web Consortium* (W3C) e que permite descrever informação de tinta digital. Contém dados como o momento de início e fim da escrita de um conjunto de *strokes*, em que cada *stroke* tem um conjunto de pontos com as coordenadas (x,y) e o respetivo *timestamp*.

Para o processamento do texto foram definidas 3 etapas: linhas, palavras e letras. As duas primeiras têm como objetivo a divisão do texto e extração de características usadas pelos investigadores. No final, tendo o texto dividido por palavras é feita a tentativa de reconhecer as letras de cada palavra, e caso exista sucesso, o reconhecimento da palavra na totalidade.

3.1.1 Linhas

Nesta fase inicial, o objetivo foi processar e retirar informações que poderão ser úteis para a análise dos investigadores e também para as fases seguintes de desconstrução do texto. Uma dessas informações é a quantidade de linhas que o texto contém. Existem algumas dificuldades em dividir um texto escrito em várias linhas, uma vez que estas podem sobrepor-se.

As duas linhas que servem de limite exterior a uma linha de texto são denominadas: *Ascent* e *Descent*. No entanto, existem ainda outras duas que são importantes para a análise; *Meanline* e *Baseline*. A Figura3.1 ilustra um exemplo das quatro linhas mencionadas.

Figura 3.1: Definição das linhas

A mais conhecida é a *baseline* que determina a linha onde as letras assentam. No caso do alfabeto latino, existem letras que descem abaixo da *baseline*, designando-se por *descent* a linha em que estas terminam. No limite superior em que a maioria das letras minúsculas são desenhadas designamos como *meanline*, tendo depois o caso das letras maiúsculas e também algumas minúsculas que passam essa linha e definimos esse limite superior como *ascent*. Esta definição será importante numa fase posterior para a análise das letras.

3.1.2 Palavras e Letras

Depois de separados os traços pertencentes a cada linha, estas são analisadas uma a uma de forma a separar os seus traços em palavras. Para a separação da linha em palavras foi utilizado o fato de que entre duas palavras existe um espaço maior que entre as letras de uma palavra. Desta forma, para a deteção de palavras foram usadas distâncias fixas.

No caso das letras a abordagem foi diferente das fases anteriores pois não é possível separar os traços das possíveis letras antes de fazer o seu reconhecimento. Em vez disso é feita uma análise aos pontos de cada palavra de forma a identificar pontos especiais

que ajudem a diferenciar as letras. Inicialmente, foram considerados especiais os pontos em que existe uma mudança vertical de direção como o exemplo da figura 3.2. Além desta característica também é guardada a direção do traço antes e depois do ponto assim como a posição vertical em que se encontra, ou seja, em que linha definida anteriormente se assenta. No entanto, depois dos testes iniciais verificamos que existem letras diferentes com os mesmos pontos especiais, pelo que se adicionou um novo tipo de ponto especial. Foi introduzida a mudança de direção horizontal, sendo o resto das características semelhantes às anteriores com a exceção da direção que passa a de ser horizontal a vertical.

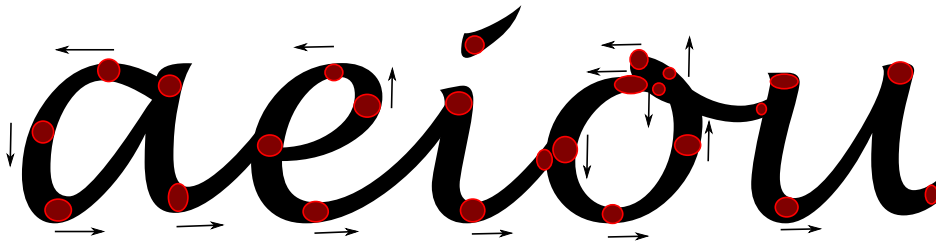


Figura 3.2: Exemplo de pontos especiais

3.2 Implementação

Para a implementação destas funcionalidades foi criada uma estrutura de dados 3.3 para descrever o texto e as suas várias partes. Esta estrutura começa pela classe *Page*, que define a página de texto a analisar e contêm toda a informação sobre este texto. Esta inclui um conjunto de linhas representadas pela classe *Line*, onde é processada e calculada toda a informação desta componente. A *Line* tem um conjunto de palavras que são representadas pela classe *Word*. Todas estas classes têm informação em comum, como a*bounding box* e o *timestamp* de inicio e fim do conjunto de *strokes*, tendo sido criada a classe *TextComponent* para representar essa informação.

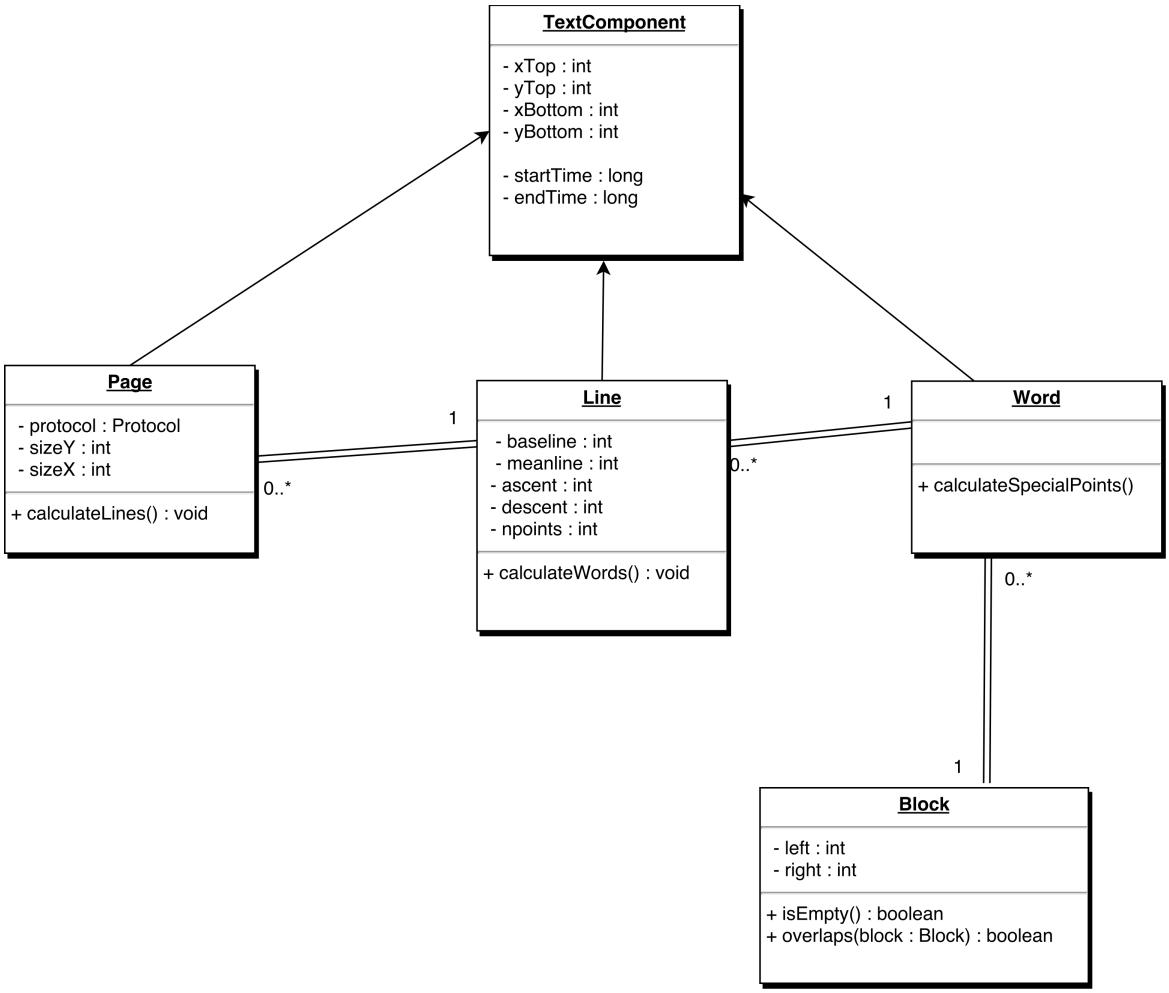


Figura 3.3: Estrutura do texto

3.2.1 Linhas

Existem dificuldades em separar as linhas de um texto manuscrito. Um exemplo é o fato dos limites *ascent* e *descent* de linhas diferentes poderem intercetar-se, como se pode ver no exemplo da figura 3.4. O primeiro passo é analisar o protocolo e contabilizar os pontos de acordo com as suas coordenadas e tendo como resultado algo semelhante ao histograma da figura 3.5. Este histograma representa a quantidade de pontos no eixo das ordenadas (y) de todo o texto de uma página.

Com esta informação é possível dividir o texto em linhas. Isto acontece porque a densidade de pontos aumenta no meio de uma linha enquanto que é quase nula nos intervalos das linhas. No entanto, existem os casos em que as linhas se intercetam e se torna mais complexo separar os traços da forma correta. Desta forma, além de analisar

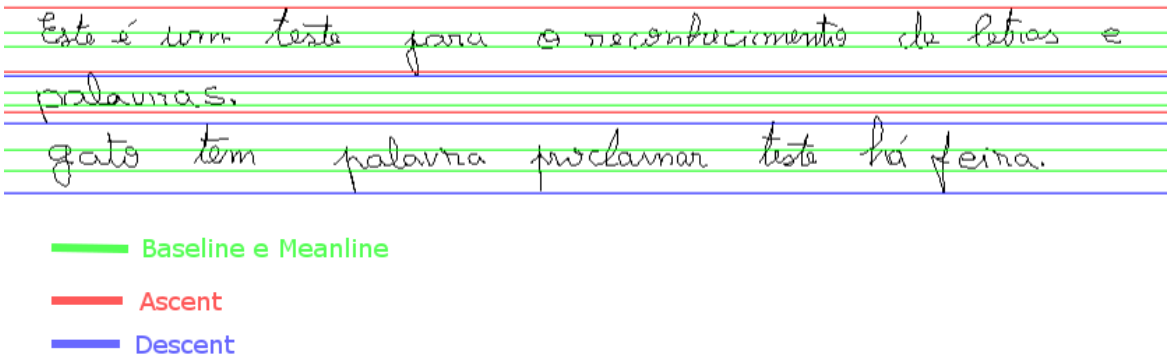


Figura 3.4: Exemplo de linhas sobrepostas

onde existe uma maior incidência de pontos é preciso analisar se o traço pertence à linha superior ou inferior.

Após a definição dos limites é necessário atribuir os traços às respetivas linhas. Nesta fase é preciso ter um especial cuidado com os casos em que existem interseções. Sendo assim, é verificado para cada traço em que linha este assenta e caso atravesse mais que uma, é calculada a percentagem de preenchimento de cada uma sendo depois o traço atribuído àquela em que tiver uma maior percentagem. Além de atribuídos os traços são eliminadas as linhas que contêm pouca densidade de pontos em relação ao resto do documento, ou seja, não contêm texto e por isso não devem ser parte da contagem das linhas e posteriormente palavras.

O último passo nesta primeira fase é a definição das quatro linhas que serão importantes para o reconhecimento de letras: *baseline*, *meanline*, *ascent* e *descent*. Para definir estas características, é utilizada a densidade dos pontos ao nível vertical em cada linha, sendo que esta será maior no espaço entre a *baseline* e a *meanline*. As linhas *descent* e *ascent* são definidas respetivamente pelos limites inferior e superior da linha. Já as outras duas linhas são mais complexas de obter. Inicialmente é encontrado o local onde existe a maior concentração de pontos e a partir daí a *baseline* e a *meanline* são deslocadas verticalmente em direções opostas, sendo recalculada a densidade de pontos abrangida pelas duas linhas. Este deslocamento para quando a densidade atinge uma percentagem superior a 50%.

3.2.2 Palavras e Letras

A separação do texto em palavras é feita linha por linha de forma a reduzir a quantidade de *strokes* a analisar de cada vez. O critério utilizado para a separação das

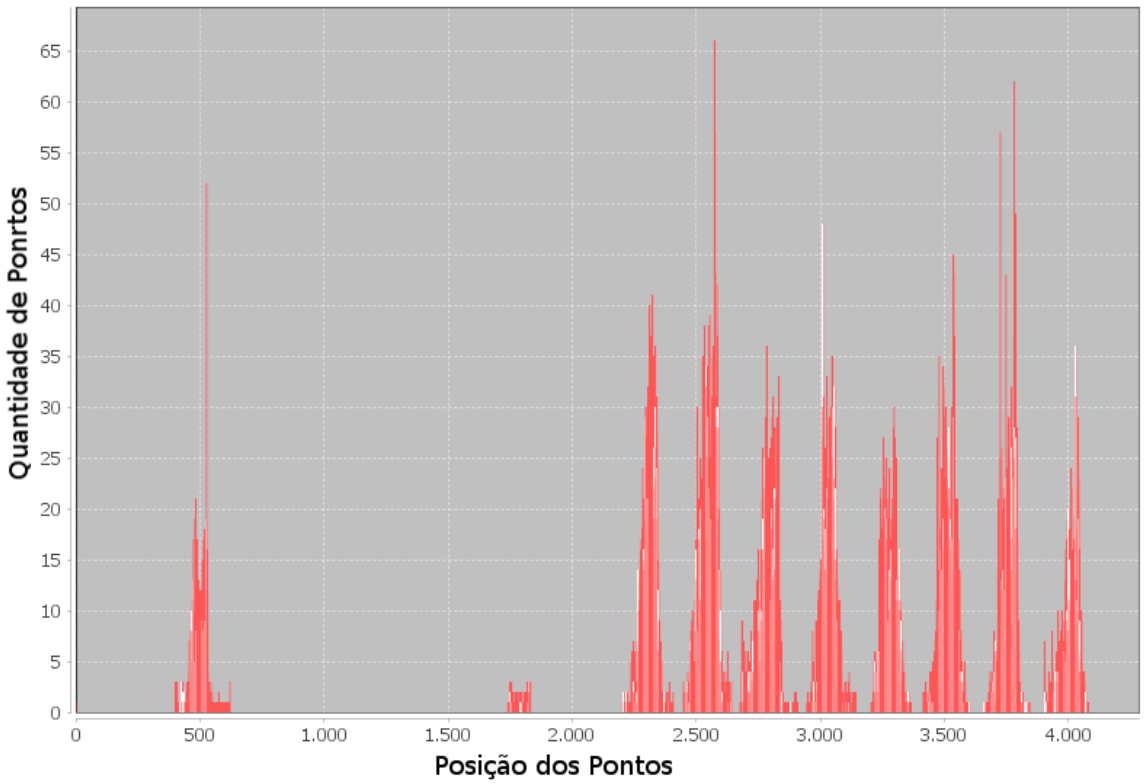


Figura 3.5: Histograma das linhas

palavras foi a ausência de pontos numa distância predefinida. Esta estratégia revelou-se demasiado simplista, não traduzindo a complexidade da separação de um texto em palavras. Isto deve-se ao fato de existirem muitas formas de escrita, desde pessoas que dentro de uma palavra deixam espaços grandes como o oposto também acontece com utilizadores a deixarem um espaço mínimo entre palavras como se pode ver na figura 3.6. É necessário criar no futuro uma forma mais dinâmica e orientada à página de definição do que é o espaço entre palavras.

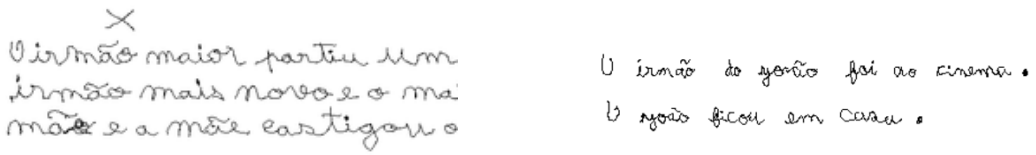


Figura 3.6: Exemplos de espaçamentos nos textos

Posteriormente, para cada palavra encontrada são analisados todos os pontos de forma a identificar e categorizar os especiais. Cada ponto especial é definido com uma direção

inicial, posição e direção final. No caso da direção esta pode ter 5 opções:

NONE Para casos em que não existe uma direção. Alguns exemplos são o início e o fim de um traço ou um ponto como o ilustrado na figura 3.7.

LEFT2RIGHT No caso dos pontos ilustrado na figura 3.8, esta opção permite definir a direção no eixo horizontal da esquerda para a direita.

RIGHT2LEFT Semelhante ao anterior mas a direção será da direita para a esquerda.

UPDOWN Para os pontos da situação da figura 3.8, esta escolha define a direção de cima para baixo em relação ao eixo vertical.

DOWNUP Da mesma forma que o anterior, permite definir a direção de baixo para cima.

No caso da posição é possível definir 7 casos:

BASELINE, MEANLINE, ASCENT e DESCENT O ponto é definido com a posição da linha que fica mais perto da posição em que se encontra.

BASELINEMEANLINE, MEANLINEASCENT, DESCENTBASELINE Nos casos em que o ponto se encontre à mesma distância de duas linhas, é definido o respetivo tipo de posição com o nome das linhas em que se encontra.

Figura 3.7: Exemplo de pontos considerados *NONE*

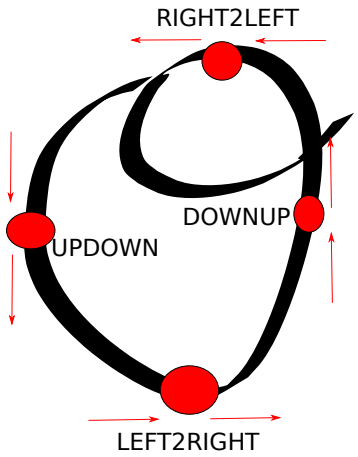


Figura 3.8: Exemplo de pontos considerados *LEFT2RIGHT*, *RIGHT2LEFT*, *UPDOWN* e *DOWNUP*

Estes pontos especiais são agrupados em blocos, em que cada bloco corresponde a um traço completo, ou seja, até que a caneta digital tenha sido levantada. Estes conjuntos de blocos são pesquisados num dicionário para o reconhecimento de letras. O dicionário de letras tem uma estrutura específica para permitir uma busca mais rápida. É criado através da análise de vários textos e contém uma árvore de prefixos (*trie*) com as letras existentes. Cada entrada no dicionário tem uma lista de pontos especiais (que correspondem a um traço). A árvore é formada através das possibilidades de combinações de pontos especiais do primeiro elemento de cada letra como se pode ver na figura 3.9. Depois de uma combinação ser correspondida por um bloco, é analisado se as possíveis letras contêm mais grupos de pontos especiais e é feita uma tentativa de correspondência com os blocos seguintes. As letras com conjuntos de pontos especiais maiores têm prioridade sobre as mais pequenas. Sempre que uma nova letra é adicionada ao dicionário, é verificado se já existe alguma letra com os mesmo conjuntos de pontos especiais. Caso exista e seja a mesma letra, esta simplesmente não é adicionada. No entanto, se existir uma letra com os mesmos pontos e esta é diferente da que está a ser adicionada é gerada uma exceção com a identificação das duas letras em conflito. Isto permite numa primeira análise verificar que letras podem gerar ambiguidade no reconhecimento, de forma a no futuro ser possível criar estratégias para lidar com estes casos.

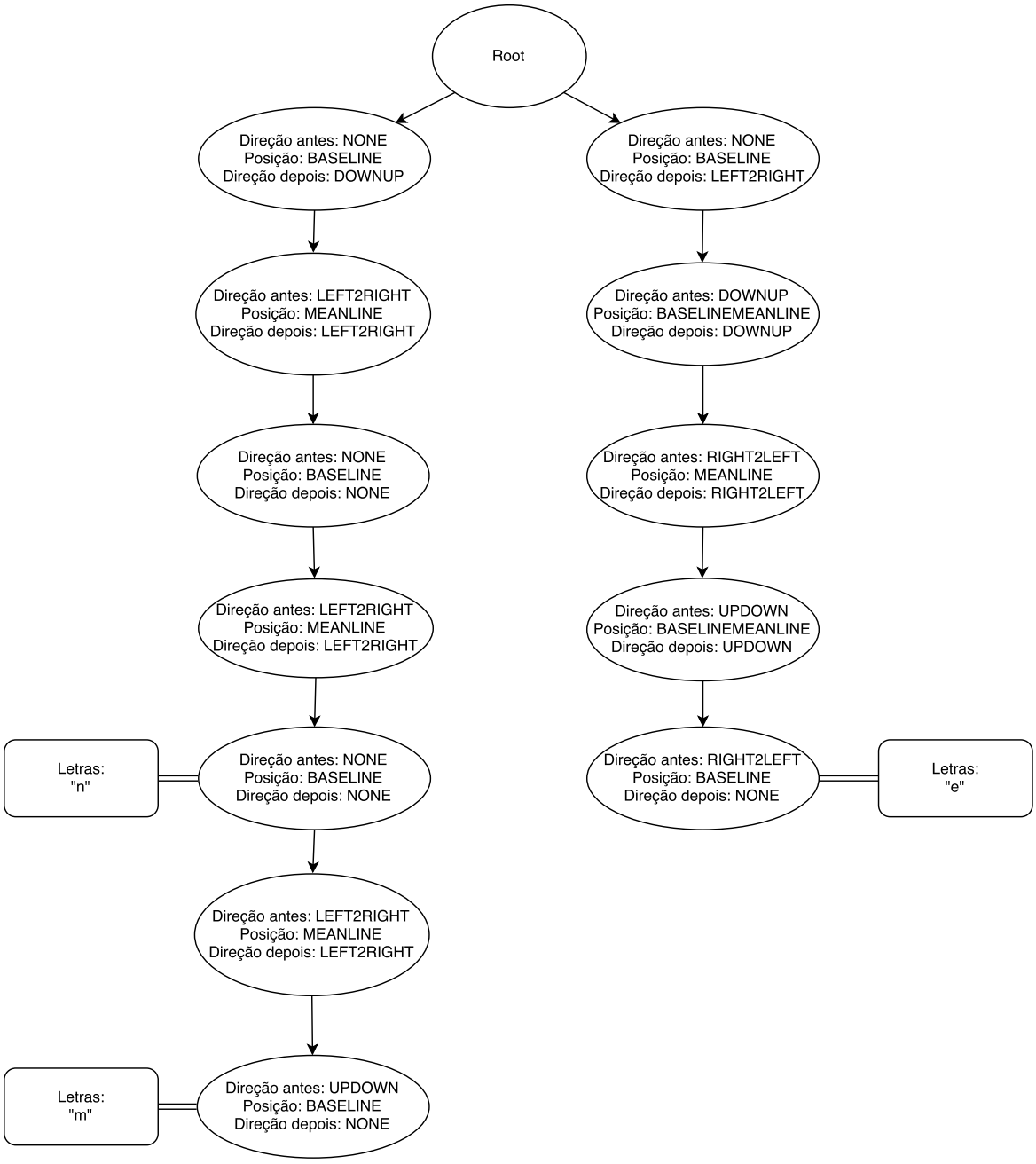


Figura 3.9: Exemplos da estrutura de uma árvore de prefixos

3.3 Validação

Esta secção tem como validar a abordagem descrita no capítulo anterior, descrevendo os testes realizados e os respetivos resultados assim como a sua interpretação para trabalhos futuros.

Para verificar qual a capacidade do programa criado para o reconhecimento de linhas, palavras e letras foram feitos vários tipos de testes. A maioria dos dados utilizados para teste foram retirados da plataforma HandSpy, tendo apenas num dos testes necessária a criação de mais dados que não existiam na plataforma. A maioria dos dados, são portanto, textos escritos por crianças de 2º ao 4º ano de escolaridade. Isto trouxe um desafio maior devido à inconsistência da forma de escrever por serem crianças numa fase inicial de aprendizagem.

Um dos testes feito foi a análise de textos de cópia, diferentes entre si mas que permitiram que se soubesse de antemão o que iria conter o texto. Neste teste é analisada a contagem de linhas, palavras e o reconhecimento de letras. Este último é feito através da contagem de letras que foram reconhecidas corretamente. Também foi contabilizado o número total de letras reconhecidas, mesmo que incorretamente, de forma a verificar a discrepância com o número real de letras do texto.

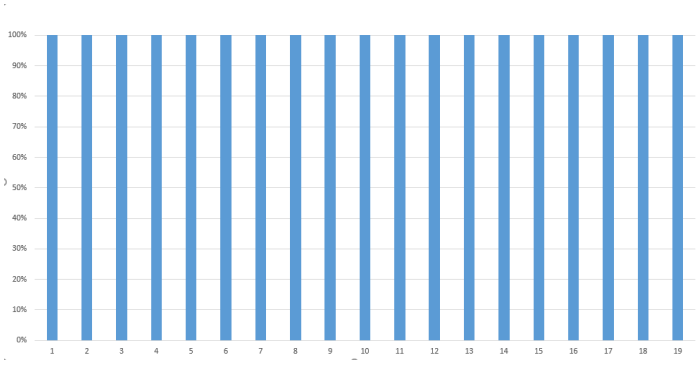


Figura 3.10: Número de linhas reconhecidas

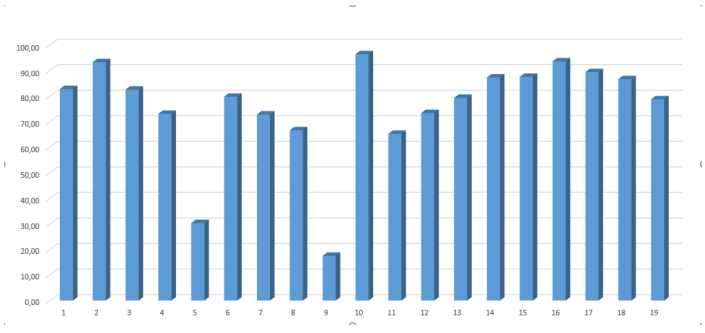


Figura 3.11: Número de palavras reconhecidas

No primeiro teste foi possível verificar que o número de linhas foi contabilizado corretamente (figura 3.10), contrastando com o número de palavras que apresentou valores relativamente baixos ao esperado inicialmente (figura 3.11). Estes valores deveram-se ao fato de ter sido utilizado um valor fixo para definir o intervalo entre

palavras e também pelo fato de não estarem a ser retiradas as rasuras e outros traços insignificantes para a contagem de palavras.

No caso do reconhecimento de letras, foi contabilizado o número total de letras reconhecido assim como a percentagem desse total que correspondia à letra a reconhecer. De notar que nos primeiros testes o dicionário de letras ainda não continha nenhum dado sobre nenhuma letra. Por isso, o valor esperado no primeiro caso era zero e deveria ir aumentando à medida que o número de letras no dicionário aumentavam. No caso das letras reconhecidas corretamente o valor alcançado não foi satisfatório pois ficou pelos 25% de acerto nos melhores casos como se pode ver na figura 3.12. Já no número de letras encontrado independentemente de ser a letra correta foram obtidos melhores resultados tendo sido atingido nalguns casos uma percentagem na ordem dos 90% de acerto (figura 3.13).

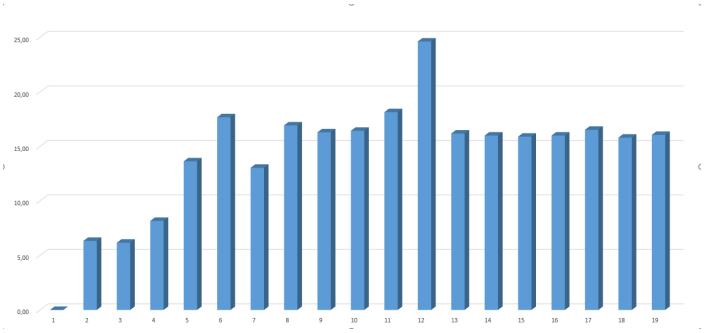


Figura 3.12: Número de letras reconhecidas corretamente

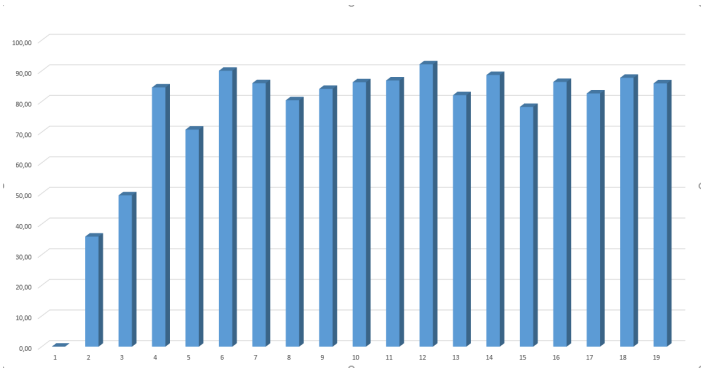


Figura 3.13: Número de letras reconhecidas independentemente de serem corretas

Apesar de não terem sido obtidos resultados satisfatórios no reconhecimento de letras, estes testes permitiram perceber a diversidade do tipo de escrita e os desafios e complexidade que isso traz ao problema. Um dos problemas detetados foi o das letras diferentes escritas da mesma forma (com a mesma sequência de pontos especiais) por

pessoas diferentes. Um exemplo são as letras "a"e "o"que têm a mesma forma em vários dos textos analisados. Outro problema detetado foram os casos em que as crianças escrevem todas as letras com a mesma altura, mesmo as que deviam atingir os limites superiores e/ou inferiores de uma linha (*ascent* e *descent*), como se pode verificar na figura 3.14. Este tipo de escrita dificulta o cálculo dos limites *baseline* e *meanline* afetando assim a definição das letras e dos respetivos pontos especiais.

(a) Letras "cho" (b) Letras "chovi" (c) Letras "partiu o vrincedo"

Figura 3.14: Exemplo de letras com a mesma altura

Capítulo 4

Conclusão

Esta tese teve como objetivo a criação de possibilidades de processamento automático de textos manuscritos de forma a prover, no futuro, mais informações sobre os textos de forma dinâmica ajudando assim os investigadores a focarem-se mais na análise dos dados gerados. Além disso, também foi feito um trabalho de manutenção da plataforma HandSpy que será descrito neste capitulo em conjunto com uma reflexão sobre o trabalho feito e as conclusões tiradas deste.

4.1 Manutenção do HandSpy

Apesar do HandSpy ser uma plataforma em funcionamento e já estar numa versão estável, ainda existiam pormenores que podiam ser melhorados. Uma das tarefas necessárias foi a correção do cálculo da distância do traço, que afetava posteriormente dados como a velocidade e a distância do *burst*. Este cálculo não estava ser feito da forma correta além de que durante a sua correção foram detetados também problemas de conversão de unidades que afetavam a correta apresentação dos dados referidos anteriormente. Aliada a esta correção esteve também o melhoramento da apresentação dos dados relacionados com a distância, que se apresentava em *Anoto Units* e passou a ser apresentada em milímetros. Inicialmente, a correção do cálculo da distância ficou operacional apenas nos novos protocolos adicionados, visto que exigia alterações nos dados dos projetos que os utilizadores poderiam não querer efetuar. Posteriormente, foi criada a opção para atualização das distâncias nos projetos antigos com a adição de um botão que recalcula as distâncias do(s) protocolo(s) selecionado(s) como se pode ver na figura4.1.

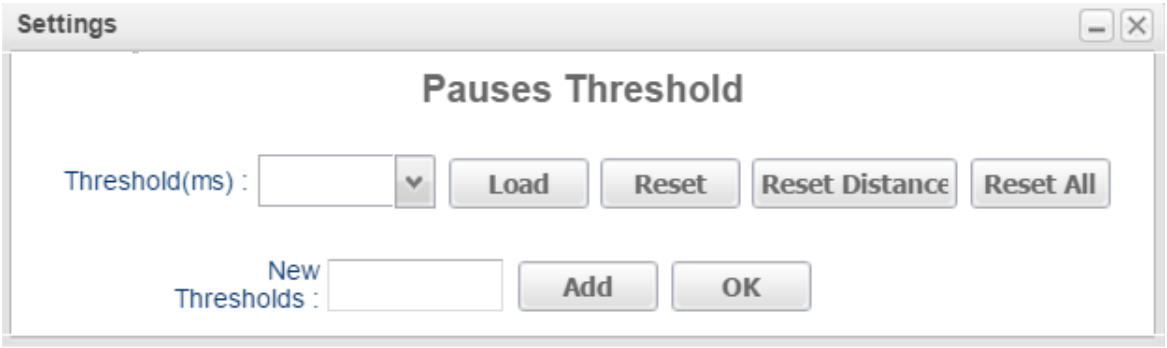


Figura 4.1: Opção de correção da distância em projetos antigos

Na página inicial de autenticação, além do *username* e da *password*, o utilizador também seleciona o projeto no qual quer entrar. Anteriormente era apresentada uma lista com todos os projetos existentes na plataforma, que com o aumento de investigadores e projetos se tornou uma lista extensa. Atualmente, esta página inicial apenas mostra os projetos do *username* inserido como se pode ver na figura 4.2.

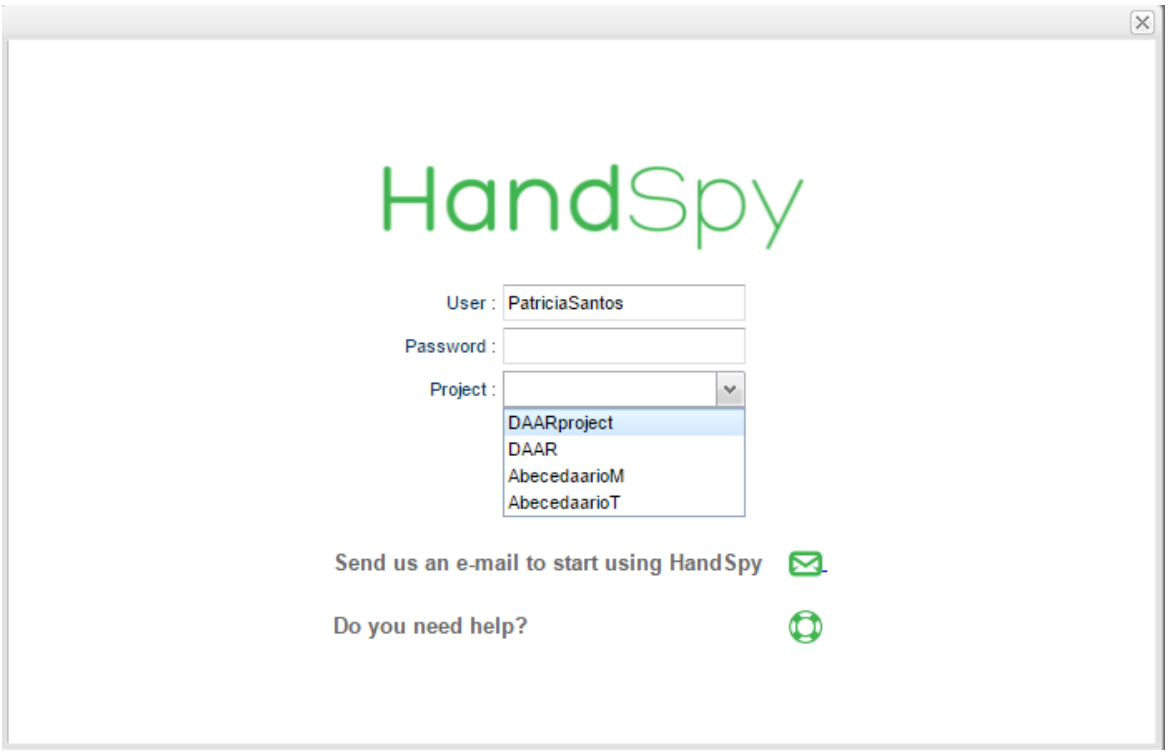


Figura 4.2: Página de *login* do HandSpy

Outra funcionalidade que não estava a funcionar corretamente era a geração de vídeos a partir dos dados dos protocolos. Estes vídeos eram gerados sem atenção ao tempo de execução da escrita, acabando por ficar demasiado rápidos. Desta forma, o código

foi reescrito de forma a gerar o vídeo com o tempo correto. No entanto, este é um processo moroso pelo a geração de vídeos *online* tem de ser feita de forma gradual e de acordo com a disponibilidade do servidor para que não consuma demasiados recursos.

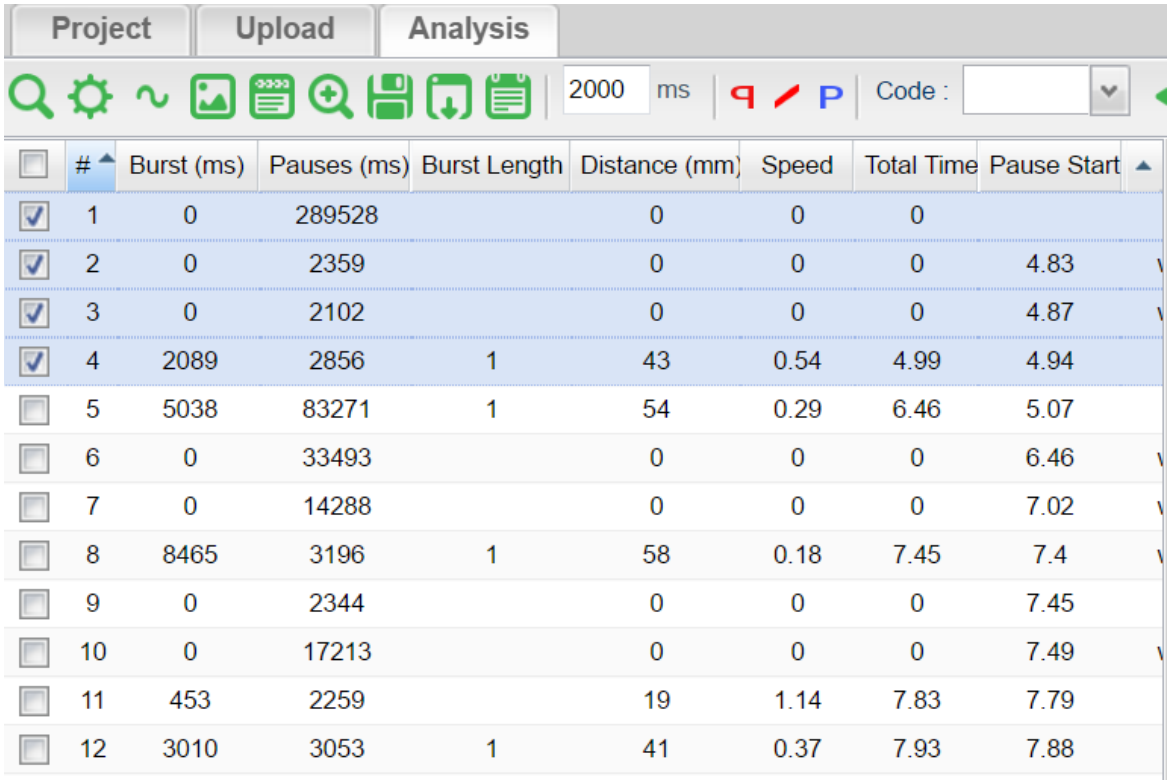
Com o aumento de projetos na plataforma, começaram a surgir situações em que o processamento era excessivamente demorado. Um desses casos foi o *upload* em grande escala de protocolos num mesmo projeto. Isto provocava o erro *HTTP 408 Request Timeout* do servidor sempre que o investigador tentava aceder à área de upload do projeto não conseguindo depois continuar o seu trabalho. A possibilidade de dividir os protocolos por vários projetos não era prático nem uma solução viável para os utilizadores. Este problema ficou resolvido com a transferência de parte do processamento para a base de dados. Sendo a base de dados XML, isto traduziu-se na criação de uma única interrogação XQuery 4.1 que substituiu vários ciclos de múltiplas interrogações feitas a partir do servidor. Atualmente, é possível fazer o carregamento de grandes quantidades de protocolos num mesmo projeto sem que este deixe de ter resposta do servidor.

```
1
2  declare namespace functx = "http://www.functx.com";
3  declare function functx:is-value-in-sequence( $value as xs:
       anyAtomicType? , $seq as xs:anyAtomicType* ) as xs:boolean
4  {
5      $value = $seq
6  };
7  let $ink := collection("/db/daar/projects/DAARproject/ink")
8  let $tasks := collection("/db/daar/projects/DAARproject/tasks")
9  let $list := (
10     for $uri in $tasks
11         let $protocol := text:groups-regex(fn:base-uri($uri),"
               /(\\d+)\\.xml$")[2]
12     return $protocol)
13
14 for $res at $pos in $ink[functx:is-value-in-sequence(text:
       groups-regex(fn:base-uri(),"/(\\d+)\\.xml$")[2],$list)]
15 return <result> <id>{text:groups-regex(fn:base-uri($res),"/(\\d
       +)\\.xml$")[2]}</id> <index>{$pos}</index> </result>
```

Listing 4.1: Exemplo de uma das XQuery implementadas

Outra otimização requerida foi na área de análise dos protocolos. Na tabela, ao

selecionar mais que uma linha (Figura 4.3) demorava alguns segundos a atualizar a imagem e os dados. O mesmo acontecia quando a seleção era removida. A solução deste caso passou por algo semelhante ao anterior, substituindo partes do código por *xqueries*, tornando a interação com a tabela mais rápida e eficiente.



#	Burst (ms)	Pauses (ms)	Burst Length	Distance (mm)	Speed	Total Time	Pause Start
1	0	289528		0	0	0	
2	0	2359		0	0	0	4.83
3	0	2102		0	0	0	4.87
4	2089	2856	1	43	0.54	4.99	4.94
5	5038	83271	1	54	0.29	6.46	5.07
6	0	33493		0	0	0	6.46
7	0	14288		0	0	0	7.02
8	8465	3196	1	58	0.18	7.45	7.4
9	0	2344		0	0	0	7.45
10	0	17213		0	0	0	7.49
11	453	2259		19	1.14	7.83	7.79
12	3010	3053	1	41	0.37	7.93	7.88

Figura 4.3: Exemplo de seleção de múltiplas linhas na análise de protocolos

Com o aumento de projetos, aumentam os pedidos de novas funcionalidades para a plataforma do HandSpy. Entre estes pedidos está a possibilidade de recolher som através das *smartpens*. As canetas utilizadas possuem um microfone e conseguem gravar som, pelo que este foi um dos desafios propostos para esta tese. No entanto, esta tarefa não acabou por ser concluída com sucesso. Isto deveu-se ao fato da empresa que produz as canetas já não disponibilizar suporte para o desenvolvimento de *software* nem existirem fóruns ou outros programadores a quem recorrer. Apesar de tudo, foram feitas várias tentativas baseadas na documentação retirada na altura em que ainda existia suporte, mas esta mostrou-se insuficiente. Tentou-se também executar a *penlet* de recolha juntamente com outra aplicação da caneta que capta o som. Infelizmente esta opção também não resultou pois as *smartpens* não conseguem executar mais que uma *penlet* ao mesmo tempo. Ou seja, ao ativar a gravação de som, deixava de gravar o texto com os dados necessários para o HandSpy. No futuro, seria ideal encontrar outras canetas que disponibilizassem esse suporte e funcionalidades.

Outro pedido feito foi a adição de novas colunas na tabela de análise. Este foi um pedido simples de executar e que permitiu a alguns investigadores adicionarem mais dados à sua análise.

4.2 Reconhecimento de Escrita

O reconhecimento de escrita manual é um desafio atual e complexo. Esta tese teve como objetivo analisar textos manuscritos e retirar destes informações que pudessem ser importantes para os investigadores que utilizam a plataforma HandSpy. Seguindo uma lógica de desconstrução de um texto em linhas e posteriormente palavras, o desenvolvimento foi feito por etapas. Começando pela divisão do texto em linhas, que acabou por tornar-se mais complexo do que inicialmente se previu pois existem casos em que as linhas se intercetam. O passo seguinte foi a separação das linhas por palavras e que demonstrou ser já de grande complexidade. O maior problema encontrado nesta fase foi como definir qual a distância que significa o inicio de uma nova palavra ou a continuação da palavra anterior. Nesta fase, foi decidido definir um valor fixo para a separação das palavras que posteriormente mostrou-se demasiado ineficaz. A última fase deste trabalho foi a tentativa de reconhecimento de letras. Esta análise era aplicada a cada palavra e os traços eram definidos com pontos considerados especiais. Através de uma interface gráfica foi possível definir manualmente que letra era definida por um conjunto especifico de pontos utilizando textos recolhidos pelos investigadores. Estas informações eram guardadas e organizadas numa *Trie* e era possível a existência de várias combinações de pontos especiais para uma letra. Ao realizar os testes com os textos manuscritos foram identificados vários problemas relacionados com a diversidade de estilos de escrita e o nível da aprendizagem da mesma. Estes fatores provocaram problemas como letras diferentes serem escritas da mesma forma por pessoas diferentes, os espaçamentos entre palavras e letras dentro da mesma palavra serem semelhantes e/ou variarem ao longo do mesmo texto. Todos estes problemas permitiram uma aprendizagem mais profunda da complexidade do reconhecimento da escrita manual e mesmo de tarefas consideradas mais básicas como a separação do texto em palavras.

Este trabalho permitiu extrair informações importantes como o número de linhas com bons resultados. No entanto, a contagem de palavras e o reconhecimento de letras ficou aquém do esperado. Apesar dos resultados menos positivos, foi iniciado um trabalho que poderá ser melhorado e utilizado posteriormente na plataforma de forma

a melhorar a experiência dos utilizadores.

4.3 Trabalho futuro

Atualmente, a informação gerada no reconhecimento de escrita ainda não está incluída na plataforma HandSpy. No futuro, o objetivo será a introdução da informação das linhas e palavras na aplicação *online* de forma a serem utilizadas e avaliadas pelos investigadores. No caso das palavras, é necessário tornar a definição da distâncias entre palavras dinâmica de forma a adaptar-se para cada tipo de texto e caligrafia. Uma possível abordagem será adaptar o espaçamento que define uma nova palavra de acordo com o documento em análise, tornando-o assim adaptado a cada tipo de escrita. Para isso terá de ser feita uma análise de todos os espaçamentos do documento e criar uma formula que dê um valor de referência para a separação das linhas em palavras. É também necessário ter em conta traços que não definem palavras como a pontuação ou zonas rasuradas. Em relação às letras, existe a necessidade de criar um plano para lidar com os casos encontrados ao longo dos testes e também a introdução de ambiguidade de forma a serem aceites letras diferentes com os mesmos pontos especiais. A escolha de qual letra deverá ser reconhecida terá depois de ter em conta o seu contexto e não apenas as características dos traços. Além disso, é preciso melhorar a ferramenta de testes a forma como é populada a base de dados de letras e pontos especiais de forma a torná-la mais dinâmica e ser possível melhora-la de forma automática.

Referências

[1] Denis Alamargot, David Chesnet, Christophe Dansac, and Christine Ros. Eye and pen: A new device for studying reading during writing. *Behavior Research Methods*, 38(2):287–299, 2006.

[2] HSM Beigi. An overview of handwriting recognition. *Proc. The 1st Annual Conference on Technological*, 1993.

[3] Line Eikvil. Optical character recognition. *citeseer. ist. psu. edu/142042. html*, 1993.

[4] Wacef Guerfali and Réjean Plamondon. Normalizing and restoring on-line handwriting. *Pattern Recognition*, 26(3):419–431, 1993.

[5] Eric Guinet and Sonia Kandel. Ductus: A software package for the study of handwriting production. *Behavior Research Methods*, 42(1):326–332, 2010.

[6] Bing Quan Huang, YB Zhang, and Mohand Tahar Kechadi. Preprocessing techniques for online handwriting recognition. In *Seventh International Conference on Intelligent Systems Design and Applications (ISDA 2007)*, pages 793–800. IEEE, 2007.

[7] Carlos Monteiro and José Paulo Leal. HandSpy - a system to manage experiments on cognitive processes in writing. In Alberto Simões, Ricardo Queirós, and Daniela da Cruz, editors, *1st Symposium on Languages, Applications and Technologies*, volume 21 of *OpenAccess Series in Informatics (OASIcs)*, pages 123–132, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[8] Carlos Monteiro and José Paulo Leal. Managing experiments on cognitive processes in writing with HandSpy. In *Computer Science and Information Systems*, volume 10, pages 1747–1773, 2013.

[9] K Nguyen-Tan and N Nguyen-Hoang. Handwriting Recognition Using B-Spline Curve. *Context-Aware Systems*, pages 335–346, 2013.

[10] A Vinciarelli. Online and offline handwriting recognition: A comprehensive survey. *Pattern Recognition*, 35(7):1433–1446, 2002.

[11] Stephen M Watt, Tom Underhill, YM Chee, K Franke, M Froumentin, S Madhvanath, JA Magaña, G Pakosz, G Russell, M Selvaraj, et al. Ink markup language (inkml). *W3C Proposed Recommendation*, 10, 2011.

Apêndice A

Acrónimos

AFD	ArisFlow Diagram
DAAR	Desenvolver, Automatizar e Autorregular os Processos Cognitivos na Composição Escrita
FCT	Fundação para a Ciência e Tecnologia
HMM	Hidden Markov Models
HTTP	Hypertext Transfer Protocol
ICR	Intelligent Character Recognition
InkML	Ink Markup Language
OCR	Optical Character Recognition
W3C	Wide Web Consortium
XML	eXtensible Markup Language
XQuery	XML Query