---

title: OCR e Tradução Automática

collection: lesson

layout: lesson

slug: OCR-e-Tradução-Automatica

date:

translation\_date:

authors:

* Andrew Akhlaghi

reviewers:

- Clemens Neudecker

- Gustavo Candela

editors:

* Anna-Maria Sichani

translator:

* Felipe Lamarca

translation-editor:

translation-reviewer:

difficulty: 2

review-ticket:

activity: transforming

topics: data manipulation

abstract: Esta lição ensinará a converter imagens de textos em ficheiros textuais, bem como a traduzir estes ficheiros. A lição também ensinará a organizar e editar imagens para tornar mais fácil e precisa a conversão e tradução de pastas inteiras de ficheiros textuais. A lição conclui com uma discussão sobre as deficiências da tradução automática e como superá-las.

---

{% include toc.html %}

# Introdução

Essa lição trata os benefícios e os desafios de se aplicarem técnicas de reconhecimento ótico de caracteres (OCR) e tradução automática em pesquisas da área de humanidades. Muitos dos meus colegas historiadores são céticos quanto a investir tempo a aprender técnicas digitais, porque não veem benefícios em suas pesquisas. Mas vivemos num mundo onde o nosso alcance digital muitas vezes ultrapassa a nossa compreensão. Investigadores podem aceder a milhares de páginas de coleções digitais online ou usar os seus próprios celulares para capturar milhares de páginas de documentos num único dia. No entanto, o acesso a esse volume e variedade de documentos também apresenta problemas. Gerenciar e organizar milhares de ficheiros de imagem não é uma tarefa fácil utilizando [interfaces gráficas de usuário]( <https://pt.wikipedia.org/wiki/Interface_gr%C3%A1fica_do_utilizador>). Além disso, ainda que o acesso aos documentos dependa menos da proximidade geográfica, a linguagem na qual o texto está escrito pode inviabilizar a sua utilização. Ter acesso aos documentos não é a mesma coisa que entendê-los. Ferramentas de linhas de comando, como simples scripts [Bash]https://pt.wikipedia.org/wiki/Bash), nos oferecem soluções para esses problemas comuns e podem tornar muito mais fácil a organização e a edição de ficheiros de imagem. A combinação de [reconhecimento ótico de caracteres (OCR)]( https://pt.wikipedia.org/wiki/Reconhecimento\_%C3%B3tico\_de\_caracteres) e [tradução automatica (APIs)]( https://pt.wikipedia.org/wiki/Tradu%C3%A7%C3%A3o\_autom%C3%A1tica), com Google Tradutor e Bing, nos promete um mundo onde qualquer texto pode ser pesquisado através de palavras-chave e traduzido. Mesmo que os programas específicos demonstrados nessa lição não sejam do seu interesse, o poder da [linguagem de script]( https://pt.wikipedia.org/wiki/Linguagem\_de\_script) ficará evidente. Combinar múltiplas ferramentas de linha de comandos e desenvolver projetos a partir delas é essencial para fazer as ferramentas digitais trabalharem para você.

Objetivos da Lição

----------------

1. Aprender como preparar documentos para OCR

2. Criar um script Bash que irá preparar, executar o OCR e traduzir todos os documentos de uma pasta

3. Aprender como criar scripts para organizar e editar os seus documentos

4. Entender as limitações do OCR e da tradução automática

# Configurações

## Linha de Comandos e Bash

-------------------------------

Este tutorial usa a [linguagem Bash de script](https://pt.wikipedia.org/wiki/Bash). Caso não esteja confortável com Bash, tire um momento para ver o tutorial do Programming Historian "[Introduction to the Bash Command Line](https://programminghistorian.org/en/lessons/intro-to-bash)" (para usuários de Linux e Mac) ou "[Introduction to the Windows Command Line with PowerShell](https://programminghistorian.org/en/lessons/intro-to-powershell)" (para usuários de Windows). A linguagem Bash já vem instalada nos sistemas operacionais Linux e Mac.

Será necessário instalar mais alguns programas. O resto desta seção ensinará como instalá-los através da linha de comandos.

Agora está na hora do nosso primeiro comando. Abra o Terminal e insira o comando `cd Desktop` para mover o seu [diretório de trabalho](https://pt.wikipedia.org/wiki/Diret%C3%B3rio\_(computa%C3%A7%C3%A3o)) para a Área de Trabalho.

## Obtenha os dados

Para este tutorial, serão utilizados dois documentos da Coleção do Arquivo Digital do Wilson Center sobre as relações entre o Irão e a União das Repúblicas Socialistas Soviéticas (URSS). Faça o download de "[Message from Bagirov to Aleksanrov on Printing Presses](https://digitalarchive.wilsoncenter.org/document/120500)" (Exemplo Um) e de "[Letter, Dimitrov to Molotov, The Situation in the Peoples's Party of Iran](https://digitalarchive.wilsoncenter.org/document/119105)" (Exemplo Dois). Esses documentos são originalmente do Arquivo de Partidos Políticos e Movimentos Sociais do Estado do Azerbaijão. Ambos estão em russo e abordam principalmente a [Crise do Irão de 1946]( https://pt.wikipedia.org/wiki/Crise\_no\_Ir%C3%A3\_de\_1946). Selecionei esses documentos por duas razões. Primeiro, porque os documentos estão escaneados em alta qualidade, mas possuem defeitos comuns a muitos documentos em acervos. Segundo, porque já possuem tradução para inglês, de modo que poderemos avaliar a qualidade/acurácia das nossas traduções automatizadas.

Salve ambos os documentos numa nova pasta na sua Área de Trabalho. De agora em diante, ir-me-ei referir aos dois documentos como Exemplo Um e Exemplo Dois. O Exemplo Um possui muito [ruído](https://pt.wikipedia.org/wiki/Ru%C3%ADdo\_digital), isto é, variações indesejadas de cor e brilho. Como é possível perceber, a imagem está distorcida e há partes escritas em diferentes fontes e tamanhos, além de marcações erradas e danos visíveis ao documento. Embora não possamos remover todos os ruídos, podemos minimizá-los com o pré-processamento da imagem.

## Processamento da Imagem

O fator que mais importa para a acurácia do OCR é a qualidade da imagem que é utilizada. Uma vez tirada a foto, é impossível melhorar a sua resolução. Além disso, uma vez que a resolução da imagem é diminuída, é impossível restaurá-la. Por isso, recomenda-se que mantenha uma cópia de cada ficheiro de imagem que utilizar. O ideal é que a cópia do ficheiro esteja no formato TIFF, porque outros formatos de ficheiro (especialmente JPG) comprimem os dados de modo que uma parte da qualidade da fotografia original é perdida. Consequentemente, ficheiros JPG são muito menores que ficheiros TIFF, o que não é necessariamente um problema. Caso esteja trabalhando com documentos datilografados que sejam claramente legíveis, não é necessário se preocupar. Mas se estiver trabalhando com documentos mais antigos, danificados ou manuscritos, provavelmente precisará que as suas imagens tenham uma resolução extra.

Ao escanear ou fotografar um documento, tenha certeza de que há luz suficiente ou que o flash está ativado, para evitar que a imagem fique muito escura (ex.: use o flash da câmara ou luzes externas adicionais) e evite fotografar de um ângulo inclinado. Em outras palavras, as linhas de texto do documento devem aparecer retas na imagem.

Frequentemente ficamos presos a imagens que possuem ruídos significativos. Por exemplo, não podemos remover danos do documento original. Há passos que podemos tomar para otimizar a imagem para OCR e melhorar a taxa de acurácia. A primeira coisa que teremos de fazer é instalar uma ferramenta gratuita de linha de comandos chamada [ImageMagick](https://imagemagick.org).

### Instalar o ImageMagick

#### Instalação no Mac

Usuários de Mac precisarão de instalar um gerenciador de pacotes chamado Homebrew. As instruções de instalação podem ser encontradas no [website do Homebrew](https://brew.sh). Para o sistema operacional do Mac, a instalação requer que entre com dois comandos simples na janela do terminal:

```brew install imagemagick```

```brew install ghostscript```

#### Instalação no Windows

As instruções de instalação do ImageMagick no Windows podem ser encontradas no [website do ImageMagick](http://imagemagick.sourceforge.net/http/www/windows.html).

### Converter PDFs em TIFFs com ImageMagick

Com o ImageMagick instalado, podemos agora converter os nossos ficheiros de PDF para TIFF e modificá-los de modo a melhorar a acurácia do OCR. Programas OCR aceitarão somente ficheiros de imagem (JPG, TIFF, PNG) como entrada, então a conversão de PDFs é necessária. O seguinte comando fará essa conversão:

```convert -density 300 NOME\_FICHEIRO\_ENTRADA.pdf -depth 8 -strip -background white -alpha off NOME\_FICHEIRO\_SAÍDA.tiff```

O comando executa várias ações que melhoram significativamente a taxa de acurácia do OCR. Os comandos `density` e `depth` garantem que o ficheiro tenha os pontos por polegada [(DPI)]( https://pt.wikipedia.org/wiki/Ponto\_por\_polegada) apropriados para OCR. Os comandos `strip`, `backgroud` e `alpha` garantem que o ficheiro tem o fundo correto. Mais importante ainda, esse comando converte o ficheiro PDF em TIFF. Ainda que não esteja utilizando um PDF, recomenda-se utilizar o comando acima para assegurar que a imagem esteja pronta para OCR.

Após essas modificações, a imagem ainda pode conter problemas. Por exemplo, é possível que haja alguma inclinação ou brilho desigual. Felizmente, o [ImageMagick](<https://imagemagick.org/index.php>) é uma ferramenta poderosa que pode-nos ajudar fazer a limpeza dos ficheiros de imagem. Para outras opções do ImageMagick que podem melhorar a qualidade do OCR, veja esta útil [coleção de scripts](http://www.fmwconcepts.com/imagemagick/textcleaner/index.php). Uma vez que o OCR é uma ferramenta da linha de comandos, é possível escrever um script que num único processo passa por todas as suas imagens (centenas e milhares) de uma vez só. Você aprenderá a escrever esse tipo de script mais a frente nessa lição.

# OCR

Esta lição utilizará o [Tesseract](https://github.com/tesseract-ocr/tesseract), o programa de execução de OCR mais popular entre os projetos de Humanidades Digitais. O Google mantem o Tesseract como um software gratuito e lançado sob a licença Apache, Versão 2.0. Este suporta mais de 100 idiomas diferentes, mas caso haja um script particularmente difícil ou único (caligrafia ou outro tipo de manuscrito), pode valer a pena treinar o seu próprio modelo de OCR. Para documentos datilografados, é necessário utilizar um programa que reconheça várias fontes similares e identifique corretamente letras imperfeitas. Tesseract 4.1 faz exatamente isso. O Google já treinou o Tesseract para reconhecer uma variedade de fontes de dezenas de idiomas. Os comandos a seguir farão a instalação do Tesseract, assim como do pacote do idioma russo, que será necessário para o resto dessa lição:

```sudo port install tesseract` `sudo port install tesseract-rus```

Pode encontrar instruções de instalação para Windows na [documentação do GitHub do Tesseract](https://github.com/UB-Mannheim/tesseract/wiki).

Os comandos para utilizar o Tesseract são relativamente simples. Basta digitar:

```tesseract NOME\_FICHEIRO\_ENTRADA NOME\_FICHEIRO\_SAÍDA -l rus```

A nossa saída tem como resultado a transcrição do ficheiro de entrada como um ficheiro de texto em russo. O parâmetro `-l` especifica o idioma de origem do documento. Outras opções de parâmetros podem ser encontradas na [documentação do GitHub do Tesseract](https://github.com/UB-Mannheim/tesseract/wiki).

# Tradução

O [Translate Shell](https://www.soimort.org/translate-shell/#translate-shell) é um programa gratuito que permite acessar a API de ferramentas de tradução automática como o [Google Tradutor]( https://translate.google.pt/?hl=pt-PT), o [Tradutor Bing]( https://www.bing.com/translator?to=pt&setlang=pt-pt), o [Yandex.Translate](https://translate.yandex.com) e o [Apertium]( https://www.apertium.org/index.por.html?dir=spa-por#?dir=por-cat&q=) através da linha de comandos, ao invés de um navegador na web. Para esse exercício, utilizaremos o Yandex em função da boa reputação desse programa em traduções do russo para o inglês e do seu alto limite de solicitações. No entanto, o Yandex não suporta tantas linguagens quanto outros tradutores. Embora as APIs de tradução não cobrem per si, elas limitam de várias formas a possibilidade de aceder através das linhas de comandos. Por exemplo, o Google Tradutor tem um limite de 5,000 caracteres por solicitação. Portanto, caso você envie ao API um ficheiro de texto de 10,000 caracteres, o Google Tradutor apenas irá traduzir os primeiros 5,000 e parar. Caso faça uma quantidade muito grande de solicitações dentro de um curto intervalo de tempo, o API bloqueará temporariamente o endereço de IP. É necessário experimentar para decidir qual API de tradução funciona melhor para cada caso.

Para instalar o Translate Shell é necessário fazer o download e rodar o pacote de instalação. Digite no terminal os comandos a seguir:

```wget git.io/trans```

e depois

```chmod +x ./trans```

Usar o Translate Shell é relativamente fácil. A linha abaixo pega num ficheiro, faz sua tradução para o inglês e salva o ficheiro resultante dessa operação.

```trans -e yandex :eng file://NOME\_FICHEIRO\_ENTRADA > NOME\_FICHEIRO\_SAÍDA```

O [parâmetro]( https://pt.wikipedia.org/wiki/Par%C3%A2metro\_(ci%C3%AAncia\_da\_computa%C3%A7%C3%A3o)) `-e` especifica o tradutor que se deseja utilizar.

# Unir tudo num único processo

Até agora passamos pelos comandos individuais de pré-processamento, execução do OCR e tradução dos nossos documentos. Essa seção cobrirá como automatizar esse processo com um script e iterar comandos sobre todos os documentos de uma pasta.

Primeiro, é necessário abrir o [Nano]( https://pt.wikipedia.org/wiki/GNU\_nano\_(editor\_de\_texto)) e começar a escrever o script. O Nano é um editor de texto gratuito e disponível nos sistemas Linux e MacOS. É simples de usar, mas tem alguns dos recursos de edição que poderíamos ver no [Emacs]( https://pt.wikipedia.org/wiki/GNU\_Emacs). Qualquer editor de texto servirá. Não é possível utilizar o seu cursor no Nano. Em vez disso, você terá de navegar utilizando as teclas de seta e `enter`. O nosso script será bem pequeno, pelo que os recursos de edição limitados do Nano não serão um problema. Quando escrever programas mais longos, recomenda-se que utilize editores de texto mais avançados. Para abrir o nano, digite `nano NOME\_DESEJADO` na linha de comandos.

Em seguida, você deve entrar em uma [shebang](https://pt.wikipedia.org/wiki/Shebang). Essa linha de comandos dirá ao computador em qual linguagem o seu script está escrito. Para um script Bash, escreva `#!/bin/bash`.

O script que iremos escrever possui três partes. Primeiro, ele solicitará que seja informada a pasta em que os ficheiros de imagem estão armazenados. Em segundo lugar, ele irá preparar, executar o OCR e traduzir as imagens desse ficheiro. Em terceiro lugar, ele irá salvar as transcrições e as traduções em ficheiros separados.

Para incorporar uma entrada do usuário, adicione `read -p` seguido por uma solicitação para o usuário. Por exemplo, as próximas duas linhas de código solicitarão que seja indicado o nome de uma pasta da sua área de trabalho e, em seguida, criarão uma variável contendo o caminho completo até essa pasta.

```

read -p "Informe o nome da pasta: " pasta;

FICHEIROS=/Users/andrewakhlaghi/Desktop/test\_dir/$pasta/\*

```

O nome da pasta que você indicar é atribuído a uma variável chamada `pasta` e, depois, é passada à variável `FICHEIROS` para completar o caminho até o ficheiro.

Em seguida, precisaremos de criar um processo de repetição para iterar sobre todos os ficheiros na pasta.

```

for f in $FICHEIROS;

do

tiff=${f%.\*}.tiff

convert -density 300 $f -depth 8 -strip -background white -alpha off $tiff

ocr=${f%.\*}\_ocr

tlate=${f%.\*}\_trans

tesseract $tiff $ocr -l rus

trans -e yandex :eng file://$ocr.txt > $tlate.txt

sleep 1m

done

```

A maior parte deste código deve ser familiar, considerando os exemplos das seções anteriores sobre ImageMagick, Tesseract e Translate Shell,. Há três adições importantes para a iteração desses processos:

1. Há uma repetição. A primeira linha cria uma nova variável, `f`, que irá receber o nome de cada ficheiro no nosso diretório.

2. Nós usamos o nome do ficheiro de imagem para criar os nomes dos ficheiros transcritos e traduzidos. O comando `${VARIABLE%.\*}` pega num ficheiro e remove a sua extensão. O comando `%` remove o sufixo. O comando `.\*` especifica que o sufixo é um "." e qualquer coisa que o acompanhar.

3. O comando `sleep 1m` impede que o programa inicie o mesmo processo para o ficheiro seguinte durante um minuto. Isso permite que o ficheiro anterior termine de ser traduzido e escrito, assim como garante que haja um espaçamento entre as requisições de tradução aos APIs, de modo a não bloquearem o IP do usuário. Talvez seja necessário ajustar o tempo de suspensão à medida que as APIs mudam as suas políticas sobre o que constituem "muitas" solicitações.

O terceiro e último bloco de código criará duas pastas para as transcrições e traduções, movendo todas as transcrições para uma pasta e todas as traduções para outra.

```

mkdir $pasta"\_ocr"

mkdir $pasta"\_traducao"

mv \*\_ocr.txt \*\_ocr

mv \*\_trans.txt \*\_traducao

```

Adicione todos os blocos juntos no seu ficheiro Nano. Lembre-se de incluir a shebang correta no início do script. Quando o ficheiro nano for salvo, é preciso torná-lo executável. Isto é, é necessário mudar as permissões no ficheiro para que ele seja tratado como um script. Para isso, insira o comando `chmod a+x NOME\_FICHEIRO`. Para executar o ficheiro escreva `./NOME\_FICHEIRO`.

# Resultados

Ao observar os resultados, verá que a tradução automatizada e o OCR requerem uma quantidade significativa de edições de alguém com conhecimento dos idiomas de origem e de destino, bem como do assunto em discussão.

Os resultados do Exemplo Um mostram o quão importante é a qualidade da imagem de entrada. A imagem deste exemplo sofre tanto de má angulação quanto de uma quantidade significativa de ruídos. A presença de manchas, listras escuras e a escrita desregular dificultam o bom desempenho do programa na classificação das letras. A inclinação torna difícil ao programa reconhecer as linhas de texto. A combinação das duas fontes de erro produz uma transcrição muito pobre.

{% include figure.html filename="OCR-e-traducao-automatica-1.png" caption="Figura 1: A nossa transcrição do Exemplo Um" %}

Os resultados do Exemplo Dois demonstram que, mesmo com uma boa imagem, a nossa transcrição e tradução iniciais ainda irão conter erros. Este possui caligrafia errónea mas, no geral, é livre de ruídos e não está inclinado. Mesmo que a conversão da imagem em texto tenha uma quantidade relativamente pequena de erros, a máquina pode não entender como traduzir todas as palavras corretamente. Por exemplo, a tradução da segunda página do Exemplo Dois contem erros de tradução, “The party’s connection to the owls.” (ver Figura 2). Esse erro vem da abreviação de “советский” (soviético), que é "COB.". Um leitor humano poderia reconhecer esse ponto como um sinal de que a palavra é uma abreviação e completar o resto da palavra levando em consideração o contexto. Embora o programa OCR tenha transcrito corretamente o ponto, o tradutor não entendeu o que fazer com ele.

{% include figure.html filename="OCR-e-traducao-automatica-2.png" caption="Figura 2: A frase com "coruja" (owl) em russo" %}

{% include figure.html filename="OCR-e-traducao-automatica-3.png" caption="Figura 3: A frase com "coruja" (“owl”) está traduzida" %}

Outro problema na tradução são os hífens. Ainda que o Tesseract tenha traduzido os hífens corretamente, nem o Tesseract nem o Yandex entenderam os seus propósitos. Embora o hífen diga ao leitor para seguir a palavra até à próxima linha, os dois programas trataram as duas metades como palavras distintas. Obviamente, é possível deletar os hífens individualmente, mas isso é moroso. Uma forma de lidar com isto é criando um pequeno script de expressões regulares (ver o tutorial do The Programming Historian ["Cleaning OCR’d Text with Regular Expressions"](https://programminghistorian.org/en/lessons/cleaning-ocrd-text-with-regular-expressions)) para deletar o hífen e unir as duas linhas.

Além dos hífens e das abreviações, o Tesseract identificou dois "a"s como "@"s na nossa frase sobre corujas (“owls”). Considerando que [e-mail]( https://pt.wikipedia.org/wiki/E-mail) não existia até o início dos anos 1960, é seguro assumir que qualquer "@" que apareça no documento é, na verdade, um "a" reconhecido incorretamente. Desse modo, podemos utilizar um script de expressões regulares ou as funções Encontrar (“Find”) e Substituir (“Replace”) do seu editor de texto para fazer as substituições.

Você também pode utilizar o comando `sed` do Bash para editar o seu documento. Por exemplo, o script `sed s/@/а/g DOCUMENTO.txt` encontrará todos os caracteres "@" e irá substituí-los por "a".

Se a frase terminar com um hífen, o script `sed` abaixo apagará todos os hífens e unirá as duas linhas:

```sed -e :a -e '/-$/N; s/-\n//; ta' ENTRADA.txt```

{% include figure.html filename="OCR-e-traducao-automatica-4.png" caption="Figura 4: A nossa mudança após algumas edições" %}

De modo bastante similar aos outros comandos mostrados anteriormente é possível manter uma lista de comandos `sed` num script mais longo e aplicá-los a outros documentos nos quais você queira executar OCR.

Após fazer as edições acima, coloque a sua transcrição editada de volta na API de tradução. Veja a melhora na frase sobre corujas (“owls”). Fica evidente como algumas edições podem melhorar radicalmente a qualidade das nossas traduções.

{% include figure.html filename="OCR-e-traducao-automatica-5.png" caption="Figura 5: A tradução melhorada" %}

# Outras Possibilidades com Scripting e ImageMagick

## Editando seus documentos com ImageMagick

Também é possível escrever um script para editar as nossas próprias imagens. Você já aprendeu como utilizar o ImageMagick para preparar um ficheiro para OCR, mas ImageMagick também possui mais opções para edição de imagens. Com relação ao Exemplo Um, três tarefas serão executadas para melhorar a acurácia do OCR:

1. Recortar a imagem e remover o excesso de espaço nas margens ao redor do documento.

2. Endireitar a imagem para que as linhas de texto fiquem paralelas à parte inferior do documento.

3. Remover todos os ruídos, especialmente as manchas escuras, que aparecem ao longo do documento.

Todas as três tarefas podem ser iteradas num script.

Os comandos de recorte serão específicos para cada documento. Há programas capazes de detetar e cortar ao redor do texto, mas esses programas inteligentes são significativamente mais complicados de se utilizar e estão fora do escopo deste tutorial. Felizmente, esses programas podem não ser necessários para editar os seus documentos. É provável que, ao fotografar documentos, seja mantida uma mesma angulação e altura em todas as fotografias, de modo que a posição relativa do texto em diferentes fotos será similar. Consequentemente, o usuário desejará aparar partes semelhantes da imagem de locais relativamente semelhantes na fotografia. Lembre-se: os documentos recortados não precisam de ser perfeitos para que o Tesseract funcione. Mas remover anotações marginais ou descolorações irá aumentar a acurácia do OCR. Após alguma experimentação perceberá que, para as imagens de entrada fornecidas no Exemplo Um, deseja-se remover 200 pixels da parte superior do documento, 250 pixels da direita, 250 pixels da parte inferior e 800 pixels da esquerda.

O script a seguir permite cortar e enquadrar todos os documentos de uma determinada pasta:

```

#!/bin/bash

read -p "Informe o nome da pasta: " pasta;

FICHEIROS=/CAMINHO\_FICHEIRO/$pasta/\*

for f in $FICHEIROS;

do

convert $f -gravity north -chop 0x200 -gravity east -chop 250x0 -gravity south -chop 0x800 -gravity west -chop 800x0 $f

convert $f -deskew 80% $f

done

```

O segundo parâmetro também irá enquadrar cada imagem. Isto é, o parâmetro `deskew` irá garantir que o corpo do texto está paralelo com o final da página. Lembre-se de que o parâmetro `chop` removerá a quantidade de pixels especificada independentemente de haver texto neles. Portanto, seja cuidadoso em relação ao conteúdo da pasta usada com este script. O script não só removerá a mesma quantidade do mesmo local em cada imagem, como também irá salvar a edição editada sobre a imagem original. Para evitar sobrescrever o original, altere o segundo nome do ficheiro `$f`. Por exemplo, se os ficheiros estivessem nomeados no formato `IMG\_XXXX.JPG`, o segundo `$f` seria substituído por `${f%.\*}\_EDITADO.jpg`. Isso removerá a extensão do nome do ficheiro de cada ficheiro e irá inserir `EDITADO.jpg` para distinguir as versões editadas.

Finalmente, podemos escrever outra seção de código para reduzir os ruídos na imagem. Como foi discutido anteriormente, ruídos são variações indesejadas no brilho e na cor da imagem. No caso do Exemplo Um, podemos ver um número grande de pontos pretos de tamanhos e formatos variados espalhados por todo o documento. Esse ruído pode ser resultado de problemas com o dispositivo de captura de imagem ou danos no documento original. O comando `despeckle` do ImageMagick detecta e reduz esses pontos. No entanto, o comando `despeckle` não possui parâmetros. Para reduzir de forma significativa o tamanho dos pontos no Exemplo Um, terá que executar o comando `despeckle` repetidas vezes no seu documento. Reescrever comandos continuamente seria entediante mas, felizmente, podemos escrever um script que repetirá o comando várias vezes.

```

#!/bin/bash

read -p "Informe o nome do ficheiro: " fl;

convert $fl -despeckle -despeckle -despeckle -despeckle -despeckle $fl

```

Esse script pegará o nome do ficheiro fornecido e executará a operação `despeckle` sobre ele cinco vezes. A saída substituirá o ficheiro de entrada original. Assim como antes, tenha a certeza de que você está no diretório de trabalho correto, pois o ficheiro especificado deve estar no seu diretório de trabalho.

A figura a seguir ilustra como ficará a aparência do Exemplo Um após o corte, o enquadramento e as repetidas remoções de manchas:

{% include figure.html filename="OCR-e-traducao-automatica-6.png" caption="Figura 6: A nova e melhorada versão do Exemplo Um" %}

## Organize os seus documentos

Os scripts também podem auxiliar na organização dos seus documentos. Por exemplo, problemas comuns de trabalho com acervos envolvem gerenciar e organizar os milhares de imagens tiradas durante uma viagem a um acervo. Provavelmente o maior problema é catalogar ficheiros pela localização do acervo. Câmaras digitais e smartphones atribuem às fotos um nome de ficheiro semelhante a `IMG\_XXXX.JPG`. Esse nome de ficheiro não informa de onde a foto veio nem o seu conteúdo. Em vez disso, o usuário pode querer que cada foto seja rotulada de acordo com o acervo onde foi tirada. É possível utilizar os [metadados]( https://pt.wikipedia.org/wiki/Metadados) de um ficheiro para escrever um script que renomeia ficheiros de acordo com o acervo de onde eles vieram.

O script a seguir irá comparar a data da última modificação de um ficheiro com a data de sua visita ao acervo e renomear o ficheiro a partir disso.

```

#!/bin/bash

read -p "Informe o nome do acervo: " $nome\_acervo;

read -p "Informe a data de visita ao acervo: " $visita;

ls -lt | awk '{if ($6$7==$visita) print $9}' >> list.txt

mkdir $nome\_acervo

for i in $(cat list.txt);

do

mv $i $nome\_acervo/$nome\_acervo${i:3};

done

```

O script irá renomear todos os ficheiros modificados no dia 30 de agosto com `[NOME\_ACERVO\_ENTRADA]\_XXXX.jpg` .

# Conclusão

Nenhum programa ou script único irá revolucionar a sua pesquisa. Por outro lado, aprender a como combinar uma variedade de ferramentas diferentes pode alterar radicalmente o modo como utiliza ficheiros e que tipo de ficheiros pode utilizar. Esta lição usou a linguagem Bash de script para agrupar ferramentas, mas é possível escolher entre uma variedade de linguagens de programação diferentes para criar os seus próprios fluxos de trabalho. Mais importante do que aprender a usar qualquer comando específico é aprender a conduzir a sua pesquisa para aproveitar ao máximo as ferramentas digitais.

A [página das lições](https://programminghistorian.org/en/lessons/) do Programming Historian oferece um bom panorama de quais as ferramentas disponíveis.

Saber as capacidades e limitações das ferramentas digitais ajudará a conduzir a sua pesquisa a aproveitá-las ao máximo.

Mesmo que não esteja interessado em OCR e tradução automatizada, os scripts mesmo assim são úteis. A habilidade de mover e renomear ficheiros pode ajudar a gerenciar a sua pesquisa. Linhas de comando serão pontos fundamentais de qualquer projeto de humanidades digitais. Esse artigo deu uma introdução à criação de scripts e ao fluxo de trabalho necessário para realmente começar a utilizar as ferramentas de humanidades digitais.