Utilizando o Tesseract OCR para reconhecimento de textos a mão

Lucas K. Uezima, Henrique Y. Y. Nishimoto, Rafael Barbosa

Faculdade de Computação e Informática  
Universidade Presbiteriana Mackenzie (UPM) – São Paulo, SP – Brazil

lucas.uezima@gmail.com, yudi.nishimoto@gmail.com, rafaelb\_ferreira@outlook.com

**Abstract.** Each year that goes by, the technological advances replace more and more the need for a human labor. Tasks that seemed impossible to be made by a computer in the past are a reality in a variety of industry sectors right now. Our research goal is to extract a handwritten text in Portuguese from images, using the OCR (Optical Character Recognition) for that, and, the machine learning to train the OCR and recognize, with accuracy, our group members writing.

**Resumo.** A cada ano que se passa os avanços tecnológicos substituem mais e mais a necessidade de mão de obra humana. Tarefas que outrora nos pareciam impossíveis de se ter um computador executando, hoje são uma realidade em diversos setores da indústria. Nosso objetivo de pesquisa é extrair o texto escrito a mão em português de imagens, utilizando para isso um OCR (Optical Character Recognition) e o conceito de aprendizado de máquina para treinar o OCR e reconhecer com precisão a escrita dos integrantes do grupo.

# Introdução

O OCR (Optical Character Recognition), reconhecimento óptico de caracteres, é uma tecnologia que possibilita a extração de texto de documentos como imagens, fotografias, PDF’s e diversos outros tipos de arquivos que contenham algum tipo de texto.

Essa tecnologia depende de vários fatores para se tornar mais assertiva na extração. O idioma, qualidade da imagem e o tipo de fonte das letras são os principais fatores que ajudam a conseguir bons resultados

Dentre os diversos tipos de OCR, escolhemos o Tesseract para utilizarmos em nosso trabalho de conclusão, devido a sua alta acuracidade e por possuir código aberto, facilita sua adaptação e treinamento para novas fontes e novas idiomas.

OCR’s são vastamente utilizados na indústria para automatizar processos. Combinado com algoritmos de machine learning ou API’s, ele consegue facilmente identificar documentos falsificados, realizar cadastros, preencher formulários e diversas outras funções de maneira muito mais rápida e efetiva do que operadores humanos

Nosso objetivo é treinar o Tesseract para que ele reconheça a escrita a mão em português dos integrantes do trabalho e analisar o comportamento e os resultados. Dessa forma exploraremos uma área muito mais complexa, visto que a escrita varia de pessoa para pessoa.

1. **Objetivo Geral**

O objetivo geral deste trabalho é desenvolver um sistema que extraia textos a mão, em português, a partir de imagens.

1. **Objetivo Específico**

Os objetivos específicos são: atingir um nível de acurácia bom a partir da caligrafia treinada pelo grupo, conseguir treinar o Tesseract com dados que otimizem o seu reconhecimento, otimizar o reconhecimento com textos em português, garantir um padrão de reconhecimento depois de treinado.

# 4. Referencial Teórico

## 4.1. Inteligência Artificial

A inteligência artificial pode haver várias definições, mas consiste em mecanismos computacionais que se baseiam no comportamento humano para resolução de problemas. A IA é um avanço tecnológico para a tomada de decisão de forma autônoma, baseada em padrões de dados. (McCarthy John, 2007)

As pesquisas sobre IA começaram após a Segunda Guerra Mundial, Alan Turing matemático inglês foi um dos primeiros a realizar uma palestra sobre o assunto em 1947. Turing acreditava que a realização de pesquisas sobre inteligência artificial era mais benéfica do que a criação de máquinas. (McCarthy John, 2007)

Em 1950 Alan Turing, criou o artigo “Computing Machinery And Intelligence”. Turing propôs considerar a questão “As máquinas podem pensar?” e defende a criação de um teste para analisar se o computador consegue ser mais inteligente que um ser humano. Este teste consiste em um jogo da simulação, para medir o desempenho da máquina diante a simulação do comportamento humano, de pensamento, e ser convincente de forma que possa enganar a inteligência humana. (Alan Turing, 2009)

Nesse período, John McCarthy foi um dos principais pesquisadores na área de Inteligência Artificial, foi organizado um seminário de dois meses em Dartmouth, em 1956 por McCarthy, Claude Shannon, Hyman Minsky e Nathaniel Rochester para trazer novidade e foi a primeira vez do termo Inteligência Artificial ser usado. (Dennis dos Santos, 2010)

McCarthy participou e realizou o artigo “Programs with Commons Sense “, é apresentado um programa proposto para resolução de problemas manipulando frases em linguagens formais. No momento em que uma frase imperativa é encontrada, poderá realizar ações como imprimir frases, movê-las ou reiniciar o processo básico. (McCarthy John, 1960)

Com o avanço da inteligência artificial, Arthur Lee Samuel um dos pioneiros na pesquisa sobre inteligência artificial em 1959, foi o criador do termo “Aprendizado de Máquina”, nessa época Samuel trabalhava na criação de máquinas autônomas com habilidade para aprenderem, sem ter sido programadas para isso. (Jude W Shavlik, 1990)

O aprendizado de máquina é uma maneira de descobrir padrões implícitos em dados não refinados, com esses dados, as máquinas podem usar os padrões para aprenderem por conta própria. Isto faz com que as máquinas ao invés de serem programadas para determinadas ações, utilizam algoritmos complexos para a melhor tomada de decisão. (Jude W Shavlik, 1990)

Os progressos no aprendizado de máquinas estão cada vez mais avançados, conforme os aplicativos aumentam de complexidade, e a demanda seja maior, podemos encontrar novos algoritmos de aprendizado de máquinas surgindo e sendo mais eficazes. Mas com todas essas novidades, as questões éticas devem ser apresentadas, sobre onde e quando devemos aplicar essa tecnologia. (Mitchell Tom Michael 2006)

## 4.2. Tesseract

O Tesseract é uma ferramenta de código aberto desenvolvido pela HP, Hewllet Packard, entre 1984 e 1994. Começou como um projeto de pesquisa e ganhou força como um possível complemento ao hardware e software da linha de scanners, visto que a maioria das ferramentas OCR da época estavam apenas iniciando e apresentavam muitas falhas. Nesse projeto, foram priorizadas suas capacidades de rejeição ao invés da melhora de sua precisão. E em 2005, a HP tornou o Tesseract em código aberto.(R. Smith, 2007)

## 4.3. Arquitetura do Tesseract

O Tesseract utiliza, como entrada, imagens binárias com regiões definidas de textos poligonais opcionais. O processamento segue o padrão passo a passo, embora alguma das etapas tenham sido incomuns para época. O primeiro passo é a análise de componentes conectados em que esboços do mesmo serão guardados, assim é possível detectar textos invertidos e reconhece-los facilmente como um texto preto no branco. E, foi a primeira ferramenta OCR a tratar de textos branco no preto. Nessa fase, os esboços são agrupados, por enquadramento, em *Blobs.*(R. Smith, 2007)

Os Blobs são organizados em linhas de texto, e as linhas e regiões são analisados para um argumento definido ou um texto proporcional. As linhas de texto são quebradas em palavras através das diferenças de espaçamento de caracteres. Textos de argumentos definidos são então separados em caracteres. O texto proporcional é separado em palavras utilizando espaços definidos e espaços incertos. O reconhecimento passa por dois passos, no primeiro as palavras são reconhecidas em cada turno. Cada palavra, considerada satisfatória, é então passada para um classificador adaptativo como dados treinados. Assim, o classificador adaptativo tem maiores chances de reconhecer melhor textos ao decorrer da página. Como essa melhora no classificador adaptativo pode ocorrer no final do texto, uma segunda checagem no texto é feita, onde palavras não bem reconhecidas são trabalhadas novamente. A fase final toma conta de espaços incertos e checa hipóteses alternativas para localizar textos em minúsculo.(R. Smith, 2007)

Para trabalhar com as linhas, as partes principais são filtragens dos blobs e a construção de linhas. Após as linhas serem encontradas, as bases das linhas serão definidas para que cada uma das linhas do texto seja separada. Para textos que não possuem um espaçamento bem definidos, o Tesseract soluciona esses problemas medindo o espaço em um alcance vertical limitado entre a base da linha e a média da linha. Espaços que estão pertos do limite se tornam difusos, para que uma decisão final seja feita após o reconhecimento da palavra. Se o resultado do reconhecimento de uma palavra não é satisfatório, as palavras são cortadas através de pontos em vértices côncavos. Se mesmo, depois de vários cortes, uma palavra ainda não ser satisfatória, ela é passada para o associador. O associador começa uma procura de combinações dos blobs, do melhor para a pior, com possíveis caracteres. Assim, esse esquema de corte e associação, simplifica a estrutura de dados que são necessárias para manter um gráfico de segmentação completa. (R. Smith, 2007)

O classificador do Tesseract utiliza de dados treinados para que consiga fazer a classificação e possíveis correspondências de caracteres. A utilização de um classificador adaptativo é recomendada para identificar diferentes tipos de fonte. A normalização de base de linha e altura x faz com que a distinção de caracteres em caixa alta e baixa sejam facilitadas. (R. Smith, 2007)

Com esses atributos, o Tesseract se tornou líder do mercado em termos de precisão, tendo as suas escolhas incomuns de atributos como uma vantagem. (R. Smith, 2007)

**4.4. Reconhecimento de línguas pelo Tesseract**

O Tesseract tinha sido desenvolvido para reconhecer apenas a língua inglesa, mas passaram a suportar outras línguas a partir de 2007.(R. W. Smith, 2013)

O Sistema de OCR para uma língua especifica não contribui muito para utilizar o OCR em outras línguas do mundo, já que há muitas em uso antes mesmo de incluir variantes históricas. (R. W. Smith, 2013)

O primeiro passo de decisão de um projeto para desenvolver um sistema OCR multilinguístico é a representação interna de Recognition Units (RU),ou, unidades de reconhecimentos. O RU é um formato individual que um mecanismo de OCR reconhece, é utilizado para generalização do alfabeto, conjunto de caracteres e grafemas, em que cada um já tem um significado específico. (R. W. Smith, 2013)

O Tesseract é regularmente testado em 30 línguas, utilizando um conjunto de testes criado de vários modos, como no caso das línguas baseadas no latim e o russo que utilizaram dados criados a partir da digitalização de livros e camadas de texto de PDF. (R. W. Smith, 2013)

**4.5. Utilização do Tesseract**

As primeiras tecnologias que utilizavam o OCR tinham de possuir scanners muito caros, ou, dispositivos óticos especializados, porém já é possível que esse reconhecimento seja feito através de um aparelho celular mesmo offline. Porém, a qualidade da imagem alterará muito o resultado do reconhecimento feito pelo OCR.(Mithe et al., 2013)

A acurácia dos dispositivos móveis excedeu a marca de 90%, com o potencial de ser maior caso seja feito um processamento da imagem. Esse processamento trataria de imperfeições na imagem como distorções, ou, falta de iluminação. (Ch et al., 2015)

**5. Metodologia**

Para realização do projeto será utilizado a API Tesseract que é uma OCR Engine (Optical Character Recognition), e, irá realizar o reconhecimento de textos a partir de imagens. O seu suporte ao Machine Learning e a fácil adaptação para novas línguas foram importantes decisões para sua escolha.

A escolha da linguagem de programação Python se deu pela familiaridade, facilidade na aplicação do Machine Learning, o que garante uma sintaxe mais simples e um bom suporte já que é uma das linguagens mais utilizadas para este fim. O Pytesseract é um wrapper que permite que o Tesseract seja utilizado com o Python, e, também aumenta o tipo de arquivos de imagem que podem ser utilizados.

Para que funcione corretamente, iremos inserir imagens com nossa própria caligrafia como dados de treinamento. Para criação e edição dos dados para definir quais são corretos ou não, será utilizado a ferramenta jTessBoxEditor, que irá criar um arquivo que conterá as predições e correções feitas pelo usuário para que o Tesseract tenha uma acurácia maior, evitando possíveis erros.

Após a criação dos dados, será necessário usar a ferramenta jTessBoxEditor para transformar a imagem com os dados da fonte, em uma linguagem do novo tesseract. O arquivo criado pela ferramenta tem o tipo “.tiff”. Após a geração do arquivo, será necessário criar o rotulo de treinamento, que irá gerar novos arquivos com ajuda da ferramenta e do tesseract.

Para realizar o rotulo de treinamento será necessário utilizar o tesseract e a ferramenta jTessBoxEditor para o auxílio da etapa. Nesse passo, sera apresentado os comandos “psm e oem”, psm informa os modos de segmentação de página, e o oem informa todos os modos de mecanismos de ocr.

Com a utilização dos comandos, o tesseract irá produzir às caixas delimitadores e a previsão de cada imagem do arquivo “.tiff” criado. Com a execução do comando, será criado um arquivo, dessa vez no formato “.box” e nele estará todas as imagens com uma caixa delimitadora de cada palavra no arquivo.

Será necessário comparar as imagens que estão no arquivo “.tiff”, ao abrir o arquivo na ferramenta, cada imagem já está com suas previsões e o próximo passo é consertar cada caixa delimitadora e as previsões que estão no arquivo “.box”.

Os dois arquivos corrigidos, criaremos um novo documento de texto que terá essa nova fonte, esse arquivo devera ser gravado na mesma pasta dos arquivos “.tiff e .box”

Com a geração dos arquivos rotulados, o treinamento com os dados terá início e será possível utilizar essa nova linguagem treinada usando o tesseract, ou seja, será criado um modelo de OCR capaz de ler fontes em português e ter a precisão de identificação dos caracteres escritos pelos integrantes.

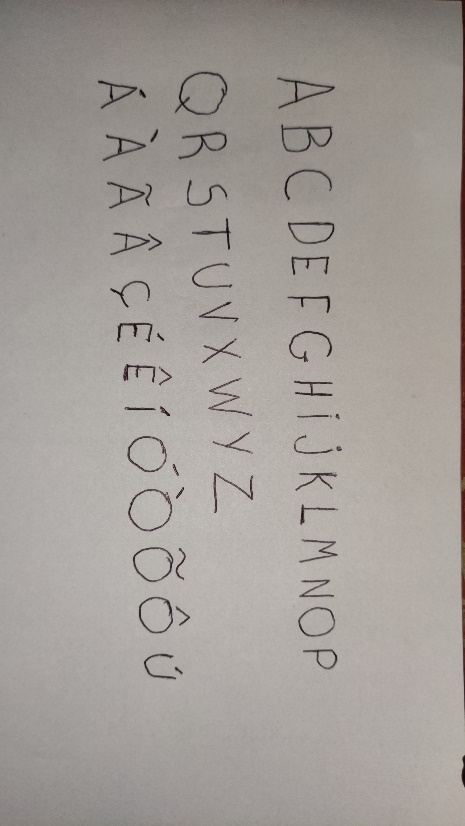
**5. Desenvolvimento e Resultados.**

Como técnica de desenvolvimento escolhemos treinar a rede neural do Tesseract com os dados de escrita em letra de forma e cursiva de um de nossos integrantes.

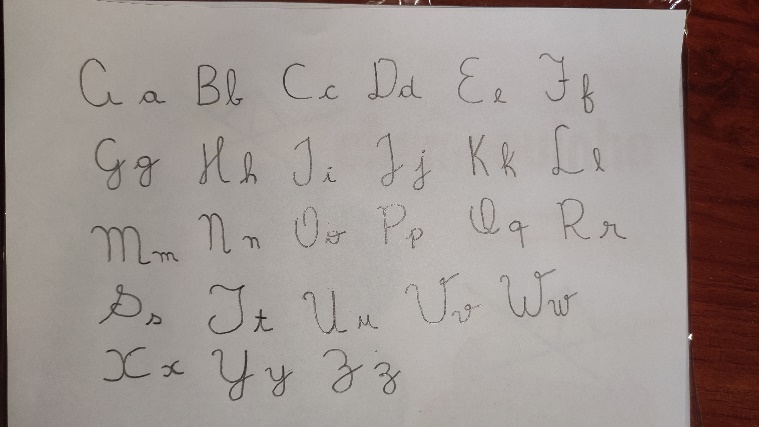
**5.1 Preparação dos dados**

Nessa etapa estudamos as técnicas de treino e separamos os dados necessários para o executar:

1. **Template de Treino:** Numa folha sulfite A4 escrevemos a mão em letra de forma (Figura 1) e cursiva(Figura 2) o alfabeto juntamente com seus caracteres especiais.

****

**Figura 1: Alfabeto em letra de mão**

****

**Figura 2: Alfabeto em letra cursiva**

1. **Tratamento do template:** Com auxílio de um software de edição melhoramos a qualidade das imagens de acordo com os requisitos de treino do tesseract. Adequamos os tamanhos dos caracteres, adicionamos a acentuação nos caracteres de letra cursiva, corrigimos o tamanho e qualidade das imagens para no mínimo 70dpi e separamos as caracteres em grupos de no máximo 16 por imagem.

**Texto, Carta

Descrição gerada automaticamente**

**Figura 3: Tratamento template letra de forma.**

**Texto, Carta

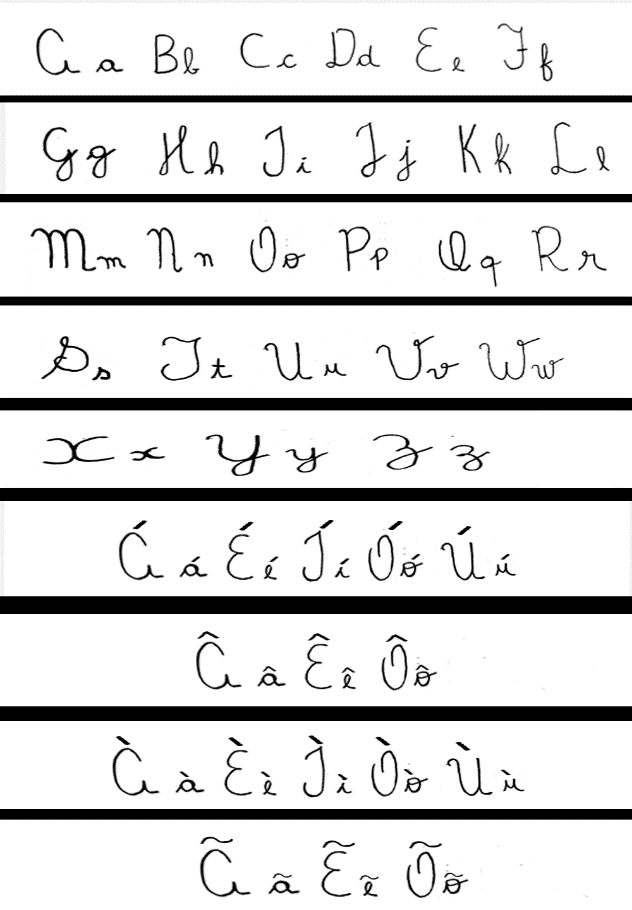
Descrição gerada automaticamente**

**Figura 4: Tratamento template letra cursiva.**

1. **Formatação do template:** Com auxílio do software *jTessBoxEditor* geramos o arquivo de treino .tiff contendo as imagens de treino de letra cursiva e letra de forma separadas em seus devidos grupos.

****

**Figura 5: Arquivo .tiff letra de forma**

****

**Figura 6: Arquivo .tif letra cursiva**

**5.2 Treino dos dados**

Para treinar a rede neural do Tesseract existem uma série de comandos e parâmetros que podem ser ou não ser utilizados. Visando agilizar o treino e retreino dos dados optamos por criar um shell script contendo todos os comandos utilizados, dessa forma automatizando e agilizando o processo.

* **Comandos & Parâmetros:** Com o arquivo .tif gerado anteriormente renomeado para a seguinte sintaxe *<idioma>.<nome>.<paginação>.tif,* utilizamos os comandos abaixo para treinar a rede neural.
  1. **tesseract --psm 6 --oem 3 por.tcc.exp0.tif por.tcc.exp0 makebox**

**-** Esse comando cria as caixas delimitadoras em cada caractere/letras e gera o arquivo .box contendo essas informações

**- --psm:** Define se a imagem contém uma palavra ou vários caracteres/letras. Utilizamos 6 pois utilizamos caracteres e não palavras para o treino

**-** **--oem**: Escolhe o método de treino. Utilizamos 3 pois é o número referente a rede neural.

* 1. **echo "por.tcc 0 0 0 0 0" > font\_properties.txt**

**-** Esse comando define as propriedades da fonte que será treinada e salva no arquivo font\_properties.txt, sendo 1 para parâmetro ativo e 0 para parâmetro desligado. Não utilizamos nenhuma propriedade pois nossa fonte é escrita de maneira manual.

**-** Os parâmetros são respectivamente:

**<fontname>:** Nome da fonte

**<italic>:** Ativar caso a fonte esteja em itálica.

**<bold>:** Ativar caso a fonte esteja em negrito.

**<fixed>:** Ativar caso a fonte seja fixada. Palavras ocupam mesmo espaço horizontalmente compõem um fonte desse tipo

**<serif>:** Ativar caso a fonte possua serifa. Serifa são hastes e prolongamentos nas letras para facilitar a leitura

**<fraktur>:** Ativar caso a fonte tenha letras quebradas em sua tipografia

* 1. **tesseract por.tcc.exp0.tif por.tcc.exp0 -l por nobatch box.train**

**-** Cria o arquivo de treino .tr com base nas informações do arquivo .box anteriormente criado

**- -l:** Define a linguagem de treino. Escolhemos “por” – português

* 1. **unicharset\_extractor por.tcc.exp0.box**

**-** Cria o arquivo que contém as propriedades dos caracteres treinados.

* 1. **shapeclustering -F font\_properties.txt -U unicharset -O por.tcc.unicharset $1.exp0.tr**

**-** Esse comando melhora a qualidade dos caracteres/letra com base nos arquivos font\_properties.txt e unicharset, e salva os dados no arquivo de treino *shapetable*

* 1. **mftraining -F font\_properties.txt -U unicharset -O por.tcc.unicharset $1.exp0.tr**
  2. **cntraining por.tcc.exp0.tr**
  3. **mv inttemp por.tcc.inttemp**

**-** Renomeia o arquivo pra inclusão do nome da fonte

* 1. **mv normproto por.tcc.normproto**

**-** Renomeia o arquivo pra inclusão do nome da fonte

* 1. **mv pffmtable por.tcc.pffmtable**

**-** Renomeia o arquivo pra inclusão do nome da fonte

* 1. **mv shapetable por.tcc.shapetable**

- Renomeia o arquivo pra inclusão do nome da fonte

* 1. **combine\_tessdata por.tcc.**

**-** Combina os arquivos de treino, (inttemp, normproto, pfftable, shatable) resultando no arquivo de treinado tessdata de nossas fontes criadas.

Texto

Descrição gerada automaticamente

**Figura 7: Script de treino em shell**

**5.3 Análise dos Resultados**

**Letra de Forma:** Tivemos bons resultados com a fonte escrita em letra de forma. A confidência do Tesseract aumentou 15% e a acurácia aumentou de 50% para 97%. Com isso

**Letra Cursiva:** Com a letra de cursiva não tivemos bons resultados, a confidência aumentou em 5% e a acurácia em 4%. Devido a escrita em letra de mão ter as palavras juntas o Tesseract não consegue diferenciar os caracteres e se perde na hora de extrair o texto.

# Referências

Ch, S., Mahna, S., & Kashyap, N. (2015). Optical Character Recognition on Handheld Devices. *International Journal of Computer Applications*, *115*(22), 10–13. https://doi.org/10.5120/20281-2833

Mithe, R., Indalkar, S., & Divekar, N. (2013). Optical character recognition. *International Journal of Recent Technology and Engineering (IJRTE)*, *2*(1), 72–75. https://doi.org/10.1177/0145482x9008401004

Smith, R. (2007). An Overview of the Tesseract OCR Engine. *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, *2*, 629–633. https://doi.org/10.1109/ICDAR.2007.4376991.

Smith, R. W. (2013). History of the Tesseract OCR engine: what worked and what didn’t. *Document Recognition and Retrieval XX*, *8658*(February 2013), 865802. https://doi.org/10.1117/12.2010051

*MCCARTHY, John. What is artificial intelligence. 2007.*

https://kewd.pw/what\_is\_artificial\_intelligence.pdf

TURING, Alan M. Computing machinery and intelligence. In: Parsing the turing test. Springer, Dordrecht, 2009. p. 23-65.

https://link.springer.com/chapter/10.1007/978-1-4020-6710-5\_3

SHAVLIK, Jude W.; DIETTERICH, Thomas; DIETTERICH, Thomas Glen (Ed.). Readings in machine learning. Morgan Kaufmann, 1990.

https://bityli.com/CSah9

GOMES, D. dos S. Inteligência Artificial: conceitos e aplicações. Olhar Científico. v1, n. 2, p. 234-246, 2010.

https://bityli.com/IdQPL

MCCARTHY, John et al. Programs with common sense. RLE and MIT computation center, 1960.

https://bityli.com/CbUxP

MITCHELL, Tom Michael. The discipline of machine learning. Pittsburgh, PA: Carnegie Mellon University, School of Computer Science, Machine Learning Department, 2006.

http://ra.adm.cs.cmu.edu/anon/usr0/ftp/anon/ml/CMU-ML-06-108.pdf