*HBI Summer Internship 2024*

**TÀI LIỆU HƯỚNG DẪN SỬ DỤNG**

**ỨNG DỤNG PHƯƠNG PHÁP DỊCH MÁY ĐỂ DỊCH TÀI LIỆU KỸ THUẬT. KỸ THUẬT XÂY DỰNG MÔ HÌNH VÀ THUẬT TOÁN DỊCH.**

# GIỚI THIỆU

Phương pháp dịch máy hay dịch máy nơ ron là phương pháp được sử dụng trong dự án này. Đối với tính chất phức tạp của một tài liệu kỹ thuật, việc sử dụng các công cụ hay phương pháp dịch thông thường sẽ mang lại kết quả không cao, các câu dịch có ý nghĩa khó hiểu và không chính xác do đó cần sử dụng phương pháp dịch nơ ron, nhờ vào khả năng học tập của mạng nơ ron các câu dịch sẽ mang văn phong của tài liệu cần dịch đó và cho kết quả dịch có ý nghĩa hơn.

Trong tài liệu này chỉ tập trung vào việc giải thích các đoạn mã Python và thuật toán của chương trình và giải thích các tham số chứ không tập trung vào cơ sở lý thuyết. Các phần của tài liệu trình bày theo các mô-đun công việc cần thực thiện để xây dựng chương trình dịch tài liệu kỹ thuật, bao gồm:

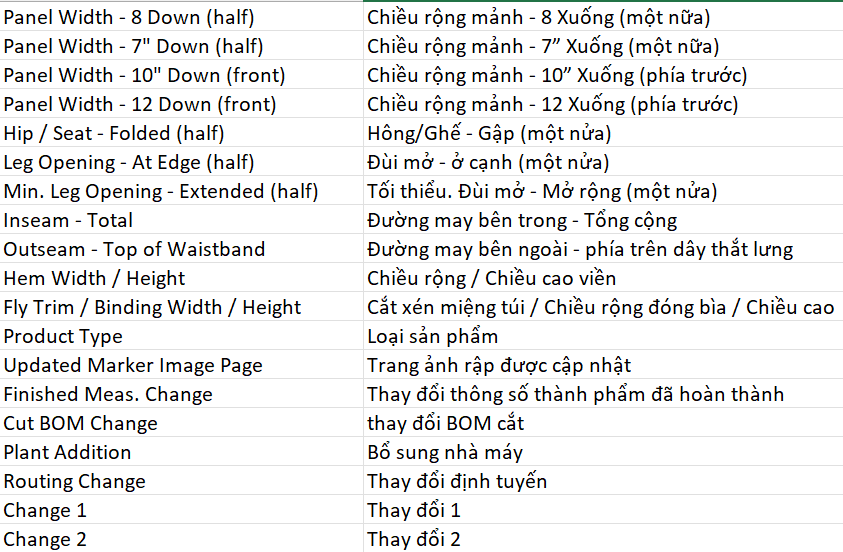
1. Xây dựng chương trình huần luyện mô hình dịch máy
2. Xây dựng chương trình đọc và dịch tài liệu

# Phần 1: Xây dựng chương trình huấn luyện mô hình dịch máy

## Chuẩn bị dữ liệu

Dữ liệu là các từ, cụm từ, câu được trích ra từ nội dung của các file tài liệu gốc và câu dịch nghĩa tương ứng của nó, nội dung dịch nghĩa có thể tùy thuộc vào văn phong của người dịch nhưng cũng cần có sự thống nhất và quy tắc nhất định trong câu dịch. Để tăng hiệu quả của việc huấn luyện, mỗi câu phức tạp (chữ, số, mã, ký tự đặc biệt) nên tạo thêm nhiều bản thể để giúp việc huấn luyện được tốt hơn, tương tự như ảnh dưới.

Dưới đây là một đoạn mẫu để chuẩn bị dữ liệu, dữ liệu có thể được lưu dưới dạng file nào tùy thuộc vào người xây dựng chương trình, trong mã của dự án này được lưu vào 2 file txt và đọc riêng biệt. Lập trình viên có thể cập nhật lại vào chung 1 file excel.



## Huấn luyện mô hình

Việc huấn luyện đòi hỏi cần có một máy tính với cấu hình cao, tuy nhiên để tiết kiệm chi phí và thuận tiện. Dự án này được triển khai huấn luyện trên nền tảng google colab. Để huấn luyện mô hình cần tạo một thư mục drive, đưa vào bộ dữ liệu đã chuẩn bị, tạo chương trình bằng google colab và lưu trữ mô hình đã huấn luyện vào đây.

Các bước xây dựng chương trình được trình bày ở các phần sau.

### Tiền xử lý dữ liệu

Dữ liệu sẽ được đọc lần lượt các câu tương ứng và mã hóa. Cần phải chèn thêm ký tự để giúp xác định điểm bắt đầu câu và điểm kết thúc câu. Và tách câu thành từng từ có nghĩa.

|  |
| --- |
| eos = '<eos>'  bos = '<bos>'  def tokenize\_en(text):      return [tok.text for tok in spacy\_en.tokenizer(text)]  from pyvi import ViTokenizer  vi\_input\_tokenize = [ViTokenizer.tokenize(i).split() for i in vi\_input]  for i in range(len(vi\_input\_tokenize)):    vi\_input\_tokenize[i].insert(0, bos)    vi\_input\_tokenize[i].insert(len(vi\_input\_tokenize[i]), eos)  import spacy  spacy\_en = spacy.load('en\_core\_web\_sm')  en\_input\_tokenize = [[tok.text for tok in spacy\_en.tokenizer(text)] for text in en\_input]  for i in range(len(en\_input\_tokenize)):    en\_input\_tokenize[i].insert(0, bos)    en\_input\_tokenize[i].insert(len(en\_input\_tokenize[i]), eos) |

Tiếp theo là bước mã hóa các câu Anh-Việt đầu vào thành các mảng số tự nhiên tương ứng. Việc mã hóa được thực hiền bằng cách tạo 2 bộ từ điển để tra cứu, cụ thể:

1. Dữ liệu tiếng Anh: mỗi từ sẽ được gắn một số tự nhiên và lưu vào một biến để làm nguồn cho mã hóa.
2. Dữ liệu tiếng Việt: tương tự.

|  |
| --- |
| from tensorflow.keras.preprocessing.text import Tokenizer  en\_tokenizer = Tokenizer()  en\_tokenizer.fit\_on\_texts(en\_input\_tokenize)  en\_vocabulary = en\_tokenizer.word\_index  en\_size = len(en\_vocabulary)  print(en\_vocabulary)  print(en\_size) # number of words in the vocabulary  vi\_tokenizer = Tokenizer()  vi\_tokenizer.fit\_on\_texts(vi\_input\_tokenize)  vi\_vocabulary = vi\_tokenizer.word\_index  vi\_size = len(vi\_vocabulary)  print(vi\_vocabulary)  print(vi\_size) |
| Kết quả:  {'<bos>': 1, '<eos>': 2, '"': 3, 'to': 4, '.': 5, 'for': 6, '-': 7, 'updated': 8,…}  749  {'<bos>': 1, '<eos>': 2, '"': 3, '.': 4, 'cập\_nhật': 5, 'chiều': 6, 'thêm': 7, …}  721 |

Sau khi tính toán, mô hình trả lại kết quả một mảng các số, từ các số này tra ngược lại từ tương ứng để tạo thành câu, do đó cũng cần một bộ để giúp tra ngược lại thuận tiện hơn.

|  |
| --- |
| en\_vocabulary\_reverse = {}  for key, value in en\_tokenizer.word\_index.items():    en\_vocabulary\_reverse[value] = key  en\_vocabulary\_reverse[0] = ''  vi\_vocabulary\_reverse = {}  for key, value in vi\_tokenizer.word\_index.items():    vi\_vocabulary\_reverse[value] = key  vi\_vocabulary\_reverse[0] = ''  print(en\_vocabulary\_reverse)  print(vi\_vocabulary\_reverse) |
| Kết quả:  {1: '<bos>', 2: '<eos>', 3: '"', 4: 'to', 5: '.', 6: 'for', 7: '-', 8: 'updated',…}  {1: '<bos>', 2: '<eos>', 3: '"', 4: '.', 5: 'cập\_nhật', 6: 'chiều', 7: 'thêm',…} |

Lưu lại các biến này để sử dụng lại:

|  |
| --- |
| import pickle  with open('/content/drive/MyDrive/hbi/dictionary\_en.pkl', 'wb') as f:      pickle.dump(en\_vocabulary, f)  with open('/content/drive/MyDrive/hbi/dictionary\_vi.pkl', 'wb') as f:      pickle.dump(vi\_vocabulary, f)  with open('/content/drive/MyDrive/hbi/dictionary\_vi\_reverse.pkl', 'wb') as f:      pickle.dump(vi\_vocabulary\_reverse, f) |

Tiếp theo là mã hóa toàn bộ dữ liệu thành các mảng có độ dài bằng nhau:

|  |
| --- |
| from tensorflow.keras.preprocessing.sequence import pad\_sequences  en\_sequence = en\_tokenizer.texts\_to\_sequences(en\_input\_tokenize)  enmaxlen = max([len(i) for i in en\_sequence])  en\_sequence = pad\_sequences(en\_sequence, maxlen = enmaxlen, padding = 'post')  print(en\_sequence[0])  vi\_sequence = vi\_tokenizer.texts\_to\_sequences(vi\_input\_tokenize)  vimaxlen = max([len(i) for i in vi\_sequence])  vi\_sequence = pad\_sequences(vi\_sequence, maxlen = vimaxlen, padding = 'post')  print(vi\_sequence[0]) |
| Kết quả:  [ 1 43 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]  [ 1 71 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0] |

Chia dữ liệu thành 2 phần: phần huấn luyện và phần để kiểm thử, kỹ thuật này là bắt buộc trong huấn luyện mô hình học máy giúp tăng độ chính xác cho mô hình được huấn luyện. Tỉ lệ chia dữ liệu và kỹ thuật để lấy dữ liệu tùy thuộc vào số lượng dữ liệu.

Trong trường hợp dữ liệu quá lớn có thể trích một phần nhỏ để sử dụng và đưa dữ liệu vào xử lý lần lược:

|  |
| --- |
| def generate\_batch(X, y, batch\_size):    while True:      for j in range(0, len(X), batch\_size):        encoder\_input\_data = []        decoder\_input\_data = []        decoder\_target\_data = []        for i, (input\_text, target\_text) in enumerate(zip(X[j:j+batch\_size], y[j:j+batch\_size])):          encoder\_input\_data.append(input\_text)          decoder\_input\_data.append(target\_text)          decodertargetdata = to\_categorical(target\_text, num\_classes=vi\_size+1)[1:]          decoder\_target\_data.append(np.concatenate((np.array(decodertargetdata), np.zeros((1, vi\_size+1))), axis = 0))        encoder\_input\_data = np.array(encoder\_input\_data)        decoder\_input\_data = np.array(decoder\_input\_data)        decoder\_target\_data = np.array(decoder\_target\_data)        yield([encoder\_input\_data, decoder\_input\_data], decoder\_target\_data) |

Tuy nhiên với dự án này, sử dụng đơn giản hơn:

|  |
| --- |
| def generate\_batch(X, y):    encoder\_input\_data = []    decoder\_input\_data = []    decoder\_target\_data = []    for i, (input\_text, target\_text) in enumerate(zip(X, y)):      encoder\_input\_data.append(input\_text)      decoder\_input\_data.append(target\_text)      decodertargetdata = to\_categorical(target\_text, num\_classes=vi\_size+1)[1:]      decoder\_target\_data.append(np.concatenate((np.array(decodertargetdata), np.zeros((1, vi\_size+1))), axis = 0))    encoder\_input\_data = np.array(encoder\_input\_data)    decoder\_input\_data = np.array(decoder\_input\_data)    decoder\_target\_data = np.array(decoder\_target\_data)    return [encoder\_input\_data, decoder\_input\_data], decoder\_target\_data |

Kỹ thuật nào thì cũng cần đảm bảo được đầu ra tương thích với đầu vào mô hình được xây dựng.

### Xây dựng và huấn luyện mô hình

Tiếp theo cần thiết lập các tham số và đầu vào cho mô hình:

|  |
| --- |
| split\_ratio = 0.8  split = round(len(vi\_sequence)\* split\_ratio)  trainX = en\_sequence[:split]  testX = en\_sequence[split:]  trainY = vi\_sequence[:split]  testY = vi\_sequence[split:]  train\_samples = len(trainX)  val\_samples = len(testX)  batch\_size = 32  epochs = 400  latent\_dim=128 |

Xây dựng mồ hình có cấu trúc như sau:

A diagram of a computer code

Description automatically generated

* Lớp encoder là lớp để mã hóa các câu tiếng Anh đầu vào thành các mảng số tự nhiên.
* Lớp decoder là lớp để giải mã các mảng đầu ra từ tính toán của mô hình thành các câu tiếng Việt tương ứng.

|  |
| --- |
| # Define an input sequence and process it.  from tensorflow.keras.layers import Input, Embedding, LSTM, Dense  from tensorflow.keras.models import Model  from tensorflow.keras.optimizers import RMSprop  from tensorflow.keras.utils import to\_categorical  import numpy as np  import re  encoder\_inputs = Input(shape=(None,))  enc\_emb = Embedding(en\_size+1, latent\_dim, mask\_zero = True)(encoder\_inputs)  encoder\_lstm = LSTM(latent\_dim, return\_state=True)  encoder\_outputs, state\_h, state\_c = encoder\_lstm(enc\_emb)  # We discard `encoder\_outputs` and only keep the states.  encoder\_states = [state\_h, state\_c]  # Set up the decoder, using `encoder\_states` as initial state.  decoder\_inputs = Input(shape=(None,))  dec\_emb\_layer = Embedding(vi\_size+1, latent\_dim, mask\_zero = True)  dec\_emb = dec\_emb\_layer(decoder\_inputs)  # We set up our decoder to return full output sequences,  # and to return internal states as well. We don't use the  # return states in the training model, but we will use them in inference.  decoder\_lstm = LSTM(latent\_dim, return\_sequences=True, return\_state=True)  decoder\_outputs, \_, \_ = decoder\_lstm(dec\_emb, initial\_state=encoder\_states)  decoder\_dense = Dense(vi\_size+1, activation='softmax')  decoder\_outputs = decoder\_dense(decoder\_outputs) |

Huấn luyện mô hình:

|  |
| --- |
| data\_train, target\_train = generate\_batch(trainX, trainY)  data\_test, target\_test = generate\_batch(testX, testY)  model = Model([encoder\_inputs, decoder\_inputs], decoder\_outputs)  model.summary()  model.compile(optimizer=RMSprop(learning\_rate=0.003), loss='categorical\_crossentropy', metrics=['acc'])  history=model.fit(data\_train, target\_train, batch\_size=batch\_size,   epochs=epochs,   validation\_data=(data\_test, target\_test), steps\_per\_epoch = train\_samples//batch\_size, validation\_steps = val\_samples//batch\_size,  ) |

Vì thời gian có hạn nên bản chất và cơ sở lý thuyết của cấu trúc mô hình không được trình bày cụ thể.

Lưu mô hình để sử dụng:

|  |
| --- |
| model.save("/content/drive/MyDrive/hbi/models/machine\_v2.keras")  !zip -r model\_keras\_v2.zip /content/drive/MyDrive/hbi/models/machine\_v2.keras  from google.colab import files  files.download ('model\_keras\_v2.zip') |

### Tải mô hình và xây dựng lớp dịch

|  |
| --- |
| import tensorflow as tf  pretrained\_model = tf.keras.models.load\_model('/content/drive/MyDrive/hbi/models/machine\_v2.keras')  # Show the model architecture  pretrained\_model.summary() |

A screenshot of a computer program

Description automatically generated

Lấy ra các lớp mô hình cần thiết để sử dụng, chú ý chính xác tên của lớp (layer):

|  |
| --- |
| pretrained\_emb\_layer = pretrained\_model.get\_layer('embedding')  encoder\_inputs = Input(shape=(None,))  encoder\_emb = pretrained\_emb\_layer(encoder\_inputs)  # Use the pre-trained layer  encoder\_lstm = pretrained\_model.get\_layer('lstm')  \_, state\_h, state\_c = encoder\_lstm(encoder\_emb)  encoder\_states = [state\_h, state\_c]  # Set up the decoder, using `encoder\_states` as initial state.  decoder\_inputs = Input(shape=(None,))  dec\_emb\_layer = pretrained\_model.get\_layer('embedding\_1')  dec\_emb = dec\_emb\_layer(decoder\_inputs)  # We set up our decoder to return full output sequences,  # and to return internal states as well. We don't use the  # return states in the training model, but we will use them in inference.  decoder\_lstm =pretrained\_model.get\_layer('lstm\_1')  decoder\_outputs, \_, \_ = decoder\_lstm(dec\_emb, initial\_state=encoder\_states)  decoder\_dense = pretrained\_model.get\_layer('dense')  decoder\_outputs = decoder\_dense(decoder\_outputs)  # Encode the input sequence to get the "Context vectors"  encoder\_model = Model(encoder\_inputs, encoder\_states)  latent\_dim=128  # Decoder setup  # Below tensors will hold the states of the previous time step  decoder\_state\_input\_h = Input(shape=(latent\_dim,))  decoder\_state\_input\_c = Input(shape=(latent\_dim,))  decoder\_state\_input = [decoder\_state\_input\_h, decoder\_state\_input\_c]  # Get the embeddings of the decoder sequence  dec\_emb2= dec\_emb\_layer(decoder\_inputs)  # To predict the next word in the sequence, set the initial states to the states from the previous time step  decoder\_outputs2, state\_h2, state\_c2 = decoder\_lstm(dec\_emb2, initial\_state=decoder\_state\_input)  decoder\_states2 = [state\_h2, state\_c2]  # A dense softmax layer to generate prob dist. over the target vocabulary  decoder\_outputs2 = decoder\_dense(decoder\_outputs2)  # Final decoder model  decoder\_model = Model(      [decoder\_inputs] + decoder\_state\_input,      [decoder\_outputs2] + decoder\_states2) |

Xây dựng hàm để dịch câu:

|  |
| --- |
| def decode\_sequence(input\_seq):      # Encode the input as state vectors.      states\_value = encoder\_model.predict(input\_seq)      # Generate empty target sequence of length 1.      target\_seq = np.zeros((1,1))      # Populate the first character of target sequence with the start character.      target\_seq[0, 0] = vi\_vocabulary[bos]      # Sampling loop for a batch of sequences (to simplify, here we assume a batch of size 1).      stop\_condition = False      decoded\_sentence = ''      # greedy search      while not stop\_condition:          output\_tokens, h, c = decoder\_model.predict([target\_seq] + states\_value)          # print(output\_tokens)          # Sample a token          sampled\_token\_index = np.argmax(output\_tokens[0, -1, :])          # print(sampled\_token\_index)          sampled\_word =vi\_vocabulary\_reverse[sampled\_token\_index]          decoded\_sentence += ' '+ sampled\_word          # Exit condition: either hit max length or find stop character.          if (sampled\_word == eos or             len(decoded\_sentence) > 300):              stop\_condition = True          # Update the target sequence (of length 1).          target\_seq = np.zeros((1,1))          target\_seq[0, 0] = sampled\_token\_index          # Update states          states\_value = [h, c]      return decoded\_sentence |

# Phần 2: Xây dựng chương trình đọc và dịch tài liệu

## Xây dựng chương trình

Các file tài liệu PDF đều được trình bày dưới dạng bảng do đó sử dụng thư viện *pymupdf,* thư viện này hỗ trợ đọc dữ liệu bảng từ PDF. Sau khi đọc dữ liệu thì ghi chúng lại vào một file excel để tiện cho quá trình dịch.

Tuy nhiên, yêu cầu đặt ra là không phải cột nào trong bảng cũng cần dịch và một số mã, ký tự, giá trị cũng không cần phải dịch do đó cần phải kiểm tra nội dung của từng ô, cột trước khi đưa vào mô hình dịch, nếu thão mãn các yêu cầu đặt ra thì được dịch còn không sẽ giữ nguyên.

Ngoài ra, đối với các câu được dịch, nếu trong câu có các ký tự đặc biệt, số, mã đặc biệt thì cần phải lưu lại và thay vào câu sau dịch để đảm bảo thông tin dịch được chính xác. Thao tác này cần thiết là do ta đang sử dụng phương pháp dịch máy, nó sẽ tính toán và dự đoán các từ sẽ xuất hiện, việc này có thể dẫn đến nó dự đoán sai các ký tự đặc biệt, số, mã.

|  |
| --- |
| def fix\_text(text):    if isinstance(text, str):      if text[:3]=='Col' or text[:3]=='Unn':        text=""      else:        text=text.replace('\n', ' ')    else:      text=""    return text  def check\_alpha(text):     if text.isalpha():       return True     else:       return False  def check\_numeric(text):      try:          float(text)          return True      except ValueError:          return False |

Hàm để dịch một câu tiếng Anh thành tiếng Việt:

|  |
| --- |
| def translate(seq\_input):    if type(seq\_input)==str:      seq\_input = seq\_input.lower()    else:      seq\_input=str(seq\_input)    seq =[tok.text for tok in spacy\_en.tokenizer(seq\_input)]    if len(seq)==1 and (not check\_alpha(seq\_input) or check\_numeric(seq\_input) or len(seq\_input)==1):      output = seq\_input    else:      list\_isalnum=[]      input\_arr = [1]      input\_arr\_=[1]      index\_next=0      enmaxlen=48 # max length of english sequence, get from source translate\_en\_vi.ipynb      for text in seq:        if text in en\_vocabulary and not (check\_numeric(text) or not check\_alpha(text) or len(text)==1):          input\_arr.append(en\_vocabulary[text])          input\_arr\_.append(en\_vocabulary[text])        else:          input\_arr.append(en\_vocabulary['<bos>'])          list\_isalnum.append(text)          try:            input\_arr\_.append(en\_vocabulary[text])          except:            input\_arr\_.append(en\_vocabulary['<bos>'])      if 'i' in list\_isalnum and 'd' in list\_isalnum:        list\_isalnum.remove('i')        list\_isalnum.remove('d')      # Translate      if np.unique(input\_arr).shape[0] == 1:        output=seq\_input      else:        input\_arr\_ = pad\_sequences([input\_arr\_], maxlen = enmaxlen, padding = 'post')        decoded\_sentence = decode\_sequence(input\_arr\_)        out\_str=decoded\_sentence[:-5].replace('\_', ' ').strip()        out\_list=out\_str.split(' ')        output=""        arr=[]        for index\_item, item in enumerate(out\_list):          if check\_alpha(item):            output+=item+" "          else:            arr.append(index\_item)        for i in list\_isalnum:          spaces\_indices = [match.start() for match in re.finditer(' ', output)]          if spaces\_indices:            try:              index=arr.pop(0)-1            except:              index=0            output=output[:spaces\_indices[index]]+" "+i+output[spaces\_indices[index]:]          else:            output += " " + i    return output |

Mỗi trang của tài liệu PDF sẽ được ghi vào một sheet của excel, và khi đưa vào dịch cũng theo từng trang, do đó cần có hàm để dịch từng trang (sheet):

|  |
| --- |
| # translator = Translator()  col\_name\_tranl=["Product Type","Piece name"]  def trans\_df(df):    name\_col=[]    list\_col=[col for col in df]    if list\_col[0] in col\_name\_tranl:      name\_col=[translate(fix\_text(col)) for col in list\_col]    else:      name\_col=[fix\_text(col) for col in list\_col]    for col in list\_col:      for row in range(len(df[col])):        df.loc[row,col]=fix\_text(df.loc[row,col])        if df.loc[row,col] in no\_translate\_vocabulary :          if df.loc[row,col][0]!="(":            df.loc[row,col]=translate(df.loc[row,col])          else:            df.loc[row,col]=df.loc[row,col]          break        else:          df.loc[row,col]=translate(df.loc[row,col])    df2=df.set\_axis(name\_col, axis=1)    return df2 |

Biến col\_name\_tranl dùng để xác định các trang nào không cần dịch. Kiểm tra ô đầu tiên của sheet, nếu chứa nội dung được xác định thì không cần phải dịch.

Chuyển toàn bộ file PDF thành excel tiếng Việt như sau:

|  |
| --- |
| doc = pymupdf.open("/content/drive/MyDrive/hbi/test.pdf")  with pd.ExcelWriter('pdf\_output.xlsx') as writer:      for index\_page in range(len(doc)):          page = doc[index\_page]          tabs = page.find\_tables(strategy='lines\_strict')          for table\_index, table in enumerate(tabs):              # Convert pymupdf Table to Pandas DataFrame              df = table.to\_pandas()              # df.to\_excel(writer, sheet\_name=f'Sheet{index\_page}\_{table\_index+1}')              result = trans\_df(df)              result.to\_excel(writer, sheet\_name=f'Sheet{index\_page}\_{table\_index+1}',index=False) |

## Sử dụng chương trình

Toàn bộ thư viện cần để chạy chương trình được lưu trong file requirements.txt.

Khi tải mã nguồn từ [github](https://github.com/ngmai1/Spec_Translator/tree/master) về thì mở terminal và cài đặt môi trường từ đây:

|  |
| --- |
| pip install -r requirements.txt |

Nhập các tài liệu cần thiết vào vị trí thư mục định sẵn và chạy file translate.py để dịch tài liệu. Kết quả là file pdf\_output.xlsx

Để huấn luyện mô hình thì sử dụng file tranning\_model.ipynb, sử dụng google colab để huấn luyện như đã trình bày ở phần trước.