```
Start coding or generate with AI.
```

Double-click (or enter) to edit

```python
# === BƯỚC 0: Thiết lập & Tải dữ liệu ===
import os, tarfile
import pandas as pd
from sklearn.preprocessing import LabelEncoder

TAR_PATH = "data/hwu.tar.gz"
if os.path.exists(TAR_PATH):
    with tarfile.open(TAR_PATH, "r:gz") as tar:
        tar.extractall("data")
    print(" Đã giải nén:", TAR_PATH)
else:
    try:
        from google.colab import files
        print(" Upload hwu.tar.gz HOẶC train/val/test (.csv)")
        up = files.upload()
        os.makedirs("data", exist_ok=True)
        for name, content in up.items():
            open(os.path.join("data", name), "wb").write(content)
        for name in up.keys():
            if name.endswith(".tar.gz") or name.endswith(".tgz"):
                with tarfile.open(os.path.join("data", name), "r:gz") as tar:
                    tar.extractall("data")
                print(" Đã giải nén:", name)
    except Exception as e:
        print(" Không chạy trong Colab hoặc upload thất bại:", e)

data_dir = "data/hwu" if os.path.isdir("data/hwu") else "data"
print(" data_dir =", data_dir, "| Files:", os.listdir(data_dir))

train_path = os.path.join(data_dir, "train.csv")
val_path   = os.path.join(data_dir, "val.csv")
test_path  = os.path.join(data_dir, "test.csv")
assert os.path.exists(train_path) and os.path.exists(val_path) and os.path.exists(test_path), \
       "Không tìm thấy train.csv/val.csv/test.csv trong " + data_dir

df_train = pd.read_csv(train_path)
df_val   = pd.read_csv(val_path)
df_test  = pd.read_csv(test_path)

def normalize_cols(df):
    cols = {c.lower(): c for c in df.columns}
    text_col = None
    for k in ["text","utterance","sentence","query","content"]:
        if k in cols: text_col = cols[k]; break
    intent_col = None
    for k in ["intent","label","category","class","target"]:
        if k in cols: intent_col = cols[k]; break
    if text_col is None: text_col = df.columns[0]
    if intent_col is None: intent_col = df.columns[1]
    return df.rename(columns={text_col:"text", intent_col:"intent"})[["text","intent"]]

df_train = normalize_cols(df_train)
df_val   = normalize_cols(df_val)
df_test  = normalize_cols(df_test)

print("Train shape:", df_train.shape)
print("Validation shape:", df_val.shape)
print("Test shape:", df_test.shape)
display(df_train.head())

le = LabelEncoder()
le.fit(pd.concat([df_train["intent"], df_val["intent"], df_test["intent"]], axis=0))

y_train = le.transform(df_train["intent"])
y_val   = le.transform(df_val["intent"])
y_test  = le.transform(df_test["intent"])
num_classes = len(le.classes_)
print(" num_classes:", num_classes, "| ví dụ nhãn:", list(le.classes_)[:10], "…")
```

```
 Upload hwu.tar.gz HOẶC train/val/test (.csv)
```
Chọn tệp Không có tệp nào được chọn Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving hwu.tar.gz to hwu.tar (6).gz
 data_dir = data/hwu | Files: ['train_10.csv', 'categories.json', 'val.csv', 'train_5.csv', 'test.csv', 'train.csv']
Train shape: (8954, 2)
Validation shape: (1076, 2)
Test shape: (1076, 2)
```

	text	intent
0	what alarms do i have set right now	alarm_query
1	checkout today alarm of meeting	alarm_query
2	report alarm settings	alarm_query
3	see see for me the alarms that you have set to...	alarm_query
4	is there an alarm for ten am	alarm_query

Nhiệm vụ 1

```python
# === NHIỆM VỤ 1: TF-IDF + Logistic Regression ===
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.metrics import classification_report, f1_score

tfidf_lr_pipeline = make_pipeline(
    TfidfVectorizer(max_features=5000),
    LogisticRegression(max_iter=1000, n_jobs=-1, random_state=42)
)

tfidf_lr_pipeline.fit(df_train["text"], y_train)
y_pred_lr = tfidf_lr_pipeline.predict(df_test["text"])

print("=== Classification report: TF-IDF + LR ===")
print(classification_report(y_test, y_pred_lr, target_names=le.classes_, digits=4))
f1_lr = f1_score(y_test, y_pred_lr, average="macro")
print("Macro-F1 (test):", f1_lr)
```

```
         datetime_query     0.7391    0.8947    0.8095        19
       email_addcontact     0.7778    0.8750    0.8235         8
            email_query     0.8333    0.7895    0.8108        19
     email_querycontact     0.9231    0.6316    0.7500        19
        email_sendemail     0.8095    0.8947    0.8500        19
         general_affirm     1.0000    1.0000    1.0000        19
    general_commandstop     1.0000    1.0000    1.0000        19
        general_confirm     1.0000    1.0000    1.0000        19
       general_dontcare     0.9048    1.0000    0.9500        19
        general_explain     1.0000    0.9474    0.9730        19
           general_joke     1.0000    1.0000    1.0000        12
         general_negate     0.9500    1.0000    0.9744        19
         general_praise     0.9500    1.0000    0.9744        19
         general_quirky     0.3571    0.2632    0.3030        19
         general_repeat     0.9048    1.0000    0.9500        19
           iot_cleaning     1.0000    1.0000    1.0000        16
             iot_coffee     1.0000    0.9474    0.9730        19
    iot_hue_lightchange     0.7500    0.7895    0.7692        19
       iot_hue_lightdim     0.9091    0.8333    0.8696        12
       iot_hue_lightoff     0.8947    0.8947    0.8947        19
        iot_hue_lighton     0.6667    0.6667    0.6667         3
        iot_hue_lightup     1.0000    0.8571    0.9231        14
           iot_wemo_off     0.8000    0.8889    0.8421         9
```

```
          qa_definition     0.8182    0.9474    0.8780        19
          qa_definition     0.8182    0.9474    0.8780        19
             qa_factoid     0.4783    0.5789    0.5238        19
               qa_maths     0.9231    0.8571    0.8889        14
               qa_stock     1.0000    0.9474    0.9730        19
   recommendation_events     0.8333    0.7895    0.8108        19
recommendation_locations     0.8095    0.8947    0.8500        19
   recommendation_movies     1.0000    1.0000    1.0000        10
            social_post     0.9500    1.0000    0.9744        19
           social_query     0.8000    0.8889    0.8421        18
         takeaway_order     0.8333    0.7895    0.8108        19
         takeaway_query     0.8947    0.8947    0.8947        19
        transport_query     0.6818    0.7895    0.7317        19
         transport_taxi     1.0000    1.0000    1.0000        18
       transport_ticket     0.9375    0.7895    0.8571        19
      transport_traffic     1.0000    0.9474    0.9730        19
          weather_query     0.6190    0.6842    0.6500        19

               accuracy                         0.8355      1076
              macro avg     0.8452    0.8343    0.8353      1076
           weighted avg     0.8422    0.8355    0.8351      1076
```

Nhiệm vụ 2

```python
# === NHIỆM VỤ 2: Word2Vec Avg + Dense ===
import numpy as np
from gensim.models import Word2Vec
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping

sentences = [str(s).split() for s in df_train["text"]]
w2v_dim = 100
w2v = Word2Vec(sentences=sentences, vector_size=w2v_dim, window=5, min_count=1, workers=4, seed=42)

def sentence_to_avg_vector(text, model, dim=100):
    toks = str(text).split()
    vecs = [model.wv[t] for t in toks if t in model.wv]
    if not vecs:
        return np.zeros(dim, dtype="float32")
    return np.mean(vecs, axis=0).astype("float32")

def to_matrix(texts, model, dim=100):
    return np.vstack([sentence_to_avg_vector(t, model, dim) for t in texts])

X_train_avg = to_matrix(df_train["text"], w2v, w2v_dim)
X_val_avg   = to_matrix(df_val["text"],   w2v, w2v_dim)
X_test_avg  = to_matrix(df_test["text"],  w2v, w2v_dim)

model_avg = Sequential([
    Dense(128, activation='relu', input_shape=(w2v_dim,)),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])
model_avg.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

es = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True, verbose=0)
_ = model_avg.fit(X_train_avg, y_train, validation_data=(X_val_avg, y_val),
                  epochs=50, batch_size=64, callbacks=[es], verbose=0)

y_pred_avg = np.argmax(model_avg.predict(X_test_avg, verbose=0), axis=1)
print("=== Classification report: W2V-Avg + Dense ===")
print(classification_report(y_test, y_pred_avg, target_names=le.classes_, digits=4))
from sklearn.metrics import f1_score
f1_avg = f1_score(y_test, y_pred_avg, average="macro")
print("Macro-F1 (test):", f1_avg)
```

```
         general_repeat     0.3333    0.3158    0.3243        19
```

```
      general_repeat     0.3333    0.3158    0.3243        19
        iot_cleaning     0.4375    0.4375    0.4375        16
          iot_coffee     0.3810    0.4211    0.4000        19
 iot_hue_lightchange     0.4091    0.4737    0.4390        19
    iot_hue_lightdim     0.4000    0.1667    0.2353        12
    iot_hue_lightoff     0.3208    0.8947    0.4722        19
     iot_hue_lighton     0.0000    0.0000    0.0000         3
     iot_hue_lightup     0.0000    0.0000    0.0000        14
        iot_wemo_off     0.0000    0.0000    0.0000         9
         iot_wemo_on     0.3333    0.2857    0.3077         7
   lists_createoradd     0.2143    0.4737    0.2951        19
         lists_query     0.0000    0.0000    0.0000        19
        lists_remove     0.2963    0.4211    0.3478        19
      music_likeness     0.0000    0.0000    0.0000        18
         music_query     0.1667    0.1053    0.1290        19
      music_settings     0.0000    0.0000    0.0000         7
          news_query     0.0870    0.1053    0.0952        19
      play_audiobook     0.0833    0.0526    0.0645        19
           play_game     0.1538    0.1053    0.1250        19
          play_music     0.0000    0.0000    0.0000        19
       play_podcasts     0.0000    0.0000    0.0000        19
          play_radio     0.0769    0.0526    0.0625        19
         qa_currency     0.2000    0.0526    0.0833        19
       qa_definition     0.0000    0.0000    0.0000        19
          qa_factoid     0.0857    0.1579    0.1111        19
            qa_maths     0.0000    0.0000    0.0000        14
            qa_stock     0.0000    0.0000    0.0000        19
recommendation_events    0.1607    0.4737    0.2400        19
recommendation_locations 0.0000    0.0000    0.0000        19
recommendation_movies    0.0000    0.0000    0.0000        10
         social_post     0.0000    0.0000    0.0000        19
        social_query     0.0000    0.0000    0.0000        18
      takeaway_order     0.1667    0.1053    0.1290        19
      takeaway_query     0.5000    0.0526    0.0952        19
     transport_query     0.0000    0.0000    0.0000        19
      transport_taxi     0.0000    0.0000    0.0000        18
    transport_ticket     0.1690    0.6316    0.2667        19
   transport_traffic     0.0000    0.0000    0.0000        19
       weather_query     0.0000    0.0000    0.0000        19

            accuracy                         0.2035      1076
           macro avg     0.1667    0.1891    0.1490      1076
        weighted avg     0.1646    0.2035    0.1557      1076

Macro-F1 (test): 0.1490183362996796
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defi
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defi
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Nhiệm vụ 3

```
# === NHIỆM VỤ 3: Embedding (pre-trained) + LSTM ===
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, LSTM
from tensorflow.keras.models import Sequential

max_words = 20000
oov_tok = "<UNK>"
tokenizer = Tokenizer(num_words=max_words, oov_token=oov_tok)
tokenizer.fit_on_texts(df_train["text"])

def to_pad(texts, tok, max_len=50):
    seqs = tok.texts_to_sequences(texts)
    return pad_sequences(seqs, maxlen=max_len, padding='post', truncating='post')

max_len = 50
X_train_pad = to_pad(df_train["text"], tokenizer, max_len)
X_val_pad   = to_pad(df_val["text"],   tokenizer, max_len)
X_test_pad  = to_pad(df_test["text"],  tokenizer, max_len)

vocab_size = min(max_words, len(tokenizer.word_index) + 1)
embedding_dim = w2v_dim

embedding_matrix = np.zeros((vocab_size, embedding_dim), dtype="float32")
for word, idx in tokenizer.word_index.items():
    if idx < vocab_size and word in w2v.wv:
        embedding_matrix[idx] = w2v.wv[word]
```

```
lstm_pre = Sequential([
    Embedding(input_dim=vocab_size, output_dim=embedding_dim,
              weights=[embedding_matrix], input_length=max_len, trainable=False),
    LSTM(128, dropout=0.2, recurrent_dropout=0.2),
    Dense(num_classes, activation='softmax')
])
lstm_pre.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

es = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True, verbose=0)
_ = lstm_pre.fit(X_train_pad, y_train, validation_data=(X_val_pad, y_val),
                 epochs=30, batch_size=64, callbacks=[es], verbose=0)

y_pred_pre = np.argmax(lstm_pre.predict(X_test_pad, verbose=0), axis=1)
print("=== Classification report: Emb(pretrained) + LSTM ===")
print(classification_report(y_test, y_pred_pre, target_names=le.classes_, digits=4))
f1_pre = f1_score(y_test, y_pred_pre, average="macro")
print("Macro-F1 (test):", f1_pre)
```

```
     general_commandstop     0.2222    0.3158    0.2609        19
         general_confirm     0.0000    0.0000    0.0000        19
        general_dontcare     0.0306    0.3684    0.0565        19
         general_explain     0.0000    0.0000    0.0000        19
            general_joke     0.0000    0.0000    0.0000        12
          general_negate     0.0820    0.2632    0.1250        19
          general_praise     0.1053    0.6316    0.1805        19
          general_quirky     0.0000    0.0000    0.0000        19
          general_repeat     0.1600    0.4211    0.2319        19
            iot_cleaning     0.2222    0.2500    0.2353        16
              iot_coffee     0.0000    0.0000    0.0000        19
     iot_hue_lightchange     0.2558    0.5789    0.3548        19
        iot_hue_lightdim     0.0000    0.0000    0.0000        12
        iot_hue_lightoff     0.4444    0.8421    0.5818        19
         iot_hue_lighton     0.0000    0.0000    0.0000         3
         iot_hue_lightup     0.0000    0.0000    0.0000        14
            iot_wemo_off     0.0000    0.0000    0.0000         9
             iot_wemo_on     0.0000    0.0000    0.0000         7
       lists_createoradd     0.0129    0.1053    0.0230        19
             lists_query     0.0000    0.0000    0.0000        19
            lists_remove     0.0000    0.0000    0.0000        19
          music_likeness     0.0000    0.0000    0.0000        18
             music_query     0.0357    0.0526    0.0426        19
          music_settings     0.0000    0.0000    0.0000         7
              news_query     0.0000    0.0000    0.0000        19
          play_audiobook     0.0000    0.0000    0.0000        19
               play_game     0.0000    0.0000    0.0000        19
              play_music     0.0000    0.0000    0.0000        19
           play_podcasts     0.0000    0.0000    0.0000        19
              play_radio     0.0000    0.0000    0.0000        19
             qa_currency     0.0000    0.0000    0.0000        19
           qa_definition     0.0000    0.0000    0.0000        19
              qa_factoid     0.0000    0.0000    0.0000        19
                qa_maths     0.0000    0.0000    0.0000        14
                qa_stock     0.0000    0.0000    0.0000        19
   recommendation_events     0.0000    0.0000    0.0000        19
recommendation_locations     0.0000    0.0000    0.0000        19
   recommendation_movies     0.0000    0.0000    0.0000        10
             social_post     0.0000    0.0000    0.0000        19
            social_query     0.0000    0.0000    0.0000        18
          takeaway_order     0.0000    0.0000    0.0000        19
          takeaway_query     0.0000    0.0000    0.0000        19
         transport_query     0.0000    0.0000    0.0000        19
          transport_taxi     0.0000    0.0000    0.0000        18
        transport_ticket     0.0000    0.0000    0.0000        19
       transport_traffic     0.0000    0.0000    0.0000        19
           weather_query     0.0000    0.0000    0.0000        19

                accuracy                         0.0818      1076
               macro avg     0.0266    0.0730    0.0362      1076
            weighted avg     0.0294    0.0818    0.0402      1076

Macro-F1 (test): 0.03616698778099441
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defi
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defi
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defi
```

Nhiệm vụ 4

```
# === NHIỆM VỤ 4: Embedding (scratch) + LSTM ===
lstm_scr = Sequential([
    Embedding(input_dim=vocab_size, output_dim=100, input_length=max_len),
    LSTM(128, dropout=0.2, recurrent_dropout=0.2),
    Dense(num_classes, activation='softmax')
])
lstm_scr.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

es = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True, verbose=0)
_ = lstm_scr.fit(X_train_pad, y_train, validation_data=(X_val_pad, y_val),
                 epochs=30, batch_size=64, callbacks=[es], verbose=0)

y_pred_scr = np.argmax(lstm_scr.predict(X_test_pad, verbose=0), axis=1)
print("=== Classification report: Emb(scratch) + LSTM ===")
print(classification_report(y_test, y_pred_scr, target_names=le.classes_, digits=4))
f1_scr = f1_score(y_test, y_pred_scr, average="macro")
print("Macro-F1 (test):", f1_scr)
```

```
     general_commandstop     0.5833    0.3684    0.4516        19
         general_confirm     0.2917    0.7368    0.4179        19
        general_dontcare     0.4412    0.7895    0.5660        19
         general_explain     0.5185    0.7368    0.6087        19
            general_joke     0.0000    0.0000    0.0000        12
          general_negate     0.0000    0.0000    0.0000        19
          general_praise     0.3333    0.4211    0.3721        19
          general_quirky     0.0000    0.0000    0.0000        19
          general_repeat     0.1250    0.0526    0.0741        19
            iot_cleaning     0.0000    0.0000    0.0000        16
              iot_coffee     0.2759    0.4211    0.3333        19
     iot_hue_lightchange     0.1923    0.5263    0.2817        19
        iot_hue_lightdim     0.0000    0.0000    0.0000        12
        iot_hue_lightoff     0.0000    0.0000    0.0000        19
         iot_hue_lighton     0.0000    0.0000    0.0000         3
         iot_hue_lightup     0.0000    0.0000    0.0000        14
            iot_wemo_off     0.0000    0.0000    0.0000         9
             iot_wemo_on     0.0000    0.0000    0.0000         7
       lists_createoradd     0.1923    0.5263    0.2817        19
             lists_query     0.0000    0.0000    0.0000        19
            lists_remove     0.2963    0.4211    0.3478        19
          music_likeness     0.0000    0.0000    0.0000        18
             music_query     0.1250    0.1579    0.1395        19
          music_settings     0.0000    0.0000    0.0000         7
              news_query     0.2333    0.3684    0.2857        19
          play_audiobook     0.2500    0.1053    0.1481        19
               play_game     0.2188    0.3684    0.2745        19
              play_music     0.1429    0.1053    0.1212        19
           play_podcasts     0.2500    0.4737    0.3273        19
              play_radio     0.2667    0.4211    0.3265        19
             qa_currency     0.1667    0.3158    0.2182        19
           qa_definition     0.1519    0.6316    0.2449        19
              qa_factoid     0.0000    0.0000    0.0000        19
                qa_maths     0.0000    0.0000    0.0000        14
                qa_stock     0.2333    0.3684    0.2857        19
   recommendation_events     0.0000    0.0000    0.0000        19
recommendation_locations     0.0000    0.0000    0.0000        19
   recommendation_movies     0.0000    0.0000    0.0000        10
             social_post     0.1538    0.2105    0.1778        19
            social_query     0.0000    0.0000    0.0000        18
          takeaway_order     0.1111    0.0526    0.0714        19
          takeaway_query     0.1714    0.3158    0.2222        19
         transport_query     0.2000    0.2105    0.2051        19
          transport_taxi     0.3750    0.1667    0.2308        18
        transport_ticket     0.2000    0.3158    0.2449        19
       transport_traffic     0.0000    0.0000    0.0000        19
           weather_query     0.0000    0.0000    0.0000        19

                accuracy                         0.2314      1076
               macro avg     0.1293    0.2049    0.1495      1076
            weighted avg     0.1458    0.2314    0.1688      1076

Macro-F1 (test): 0.14952501738646845
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defi
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defi
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defi
```

Nhiệm vụ 5

```
# === NHIỆM VỤ 5: So sánh định lượng + Phân tích định tính ===
import numpy as np
import pandas as pd
from sklearn.metrics import f1_score, classification_report

loss_avg, _ = model_avg.evaluate(X_test_avg, y_test, verbose=0)
loss_pre, _ = lstm_pre.evaluate(X_test_pad, y_test, verbose=0)
loss_scr, _ = lstm_scr.evaluate(X_test_pad, y_test, verbose=0)

summary = pd.DataFrame({
    "Pipeline": [
        "TF-IDF + Logistic Regression",
        "Word2Vec (Avg) + Dense",
        "Embedding (Pre-trained) + LSTM",
        "Embedding (Scratch) + LSTM"
    ],
    "F1-macro (test)": [f1_lr, f1_avg, f1_pre, f1_scr],
    "Test Loss": [None, loss_avg, loss_pre, loss_scr]
}).sort_values("F1-macro (test)", ascending=False).reset_index(drop=True)

display(summary)

hard_texts = [
    "can you remind me to not call my mom",
    "is it going to be sunny or rainy tomorrow",
    "find a flight from new york to london but not through paris"
]

def predict_all(texts):
    pred_lr = tfidf_lr_pipeline.predict(texts)
    Xavg = np.vstack([sentence_to_avg_vector(t, w2v, w2v_dim) for t in texts])
    pred_avg = np.argmax(model_avg.predict(Xavg, verbose=0), axis=1)
    seqs = tokenizer.texts_to_sequences(texts)
    Xpad = pad_sequences(seqs, maxlen=max_len, padding='post', truncating='post')
    pred_pre = np.argmax(lstm_pre.predict(Xpad, verbose=0), axis=1)
    pred_scr = np.argmax(lstm_scr.predict(Xpad, verbose=0), axis=1)
    return pred_lr, pred_avg, pred_pre, pred_scr

pred_lr, pred_avg, pred_pre, pred_scr = predict_all(hard_texts)

true_labels = []
for t in hard_texts:
    match = df_test[df_test["text"].str.lower() == t.lower()]
    if len(match) > 0:
        true_labels.append(le.transform(match["intent"])[0])
    else:
        true_labels.append(None)

rows = []
for i, text in enumerate(hard_texts):
    y_true = true_labels[i]
    row = {
        "text": text,
        "True Intent": le.classes_[y_true] if y_true is not None else "N/A",
        "TF-IDF + LR": le.classes_[pred_lr[i]],
        "W2V-Avg + Dense": le.classes_[pred_avg[i]],
        "LSTM (pre)": le.classes_[pred_pre[i]],
        "LSTM (scratch)": le.classes_[pred_scr[i]],
    }
    if y_true is not None:
        row["✓ LR"] = "✓" if pred_lr[i] == y_true else "✗"
        row["✓ W2V"] = "✓" if pred_avg[i] == y_true else "✗"
        row["✓ LSTM-pre"] = "✓" if pred_pre[i] == y_true else "✗"
        row["✓ LSTM-scr"] = "✓" if pred_scr[i] == y_true else "✗"
    rows.append(row)

qual_df = pd.DataFrame(rows)
display(qual_df)

print("\n Nhận xét:")
print("- Các câu có phủ định ('not') hoặc mệnh đề phụ ('or', 'but not through ...') thường khiến mô hình TF-IDF và W2V trung bìn
      "vì chúng không nắm được thứ tự và phạm vi của phủ định.")
print("- LSTM (đặc biệt bản pre-trained) có xu hướng hiểu ngữ cảnh tốt hơn, vì trạng thái ẩn theo chuỗi giúp mô hình nhận diện đ
print("- Nếu LSTM (pre-trained) chính xác hơn bản học từ đầu, điều đó chứng tỏ embedding Word2Vec cung cấp ngữ nghĩa ban đầu hữu
```

	Pipeline	F1-macro (test)	Test Loss
0	TF-IDF + Logistic Regression	0.835298	NaN
1	Embedding (Scratch) + LSTM	0.149525	2.941058
2	Word2Vec (Avg) + Dense	0.149018	3.032118
3	Embedding (Pre-trained) + LSTM	0.036167	3.603505

	text	True Intent	TF-IDF + LR	W2V-Avg + Dense	LSTM (pre)	LSTM (scratch)
0	can you remind me to not call my mom	N/A	calendar_set	general_joke	general_dontcare	email_sendemail
1	is it going to be sunny or rainy tomorrow	N/A	weather_query	calendar_query	lists_createoradd	social_post
2	find a flight from new york to london but not ...	N/A	general_negate	transport_query	alarm_set	cooking_recipe

```
 Nhận xét:
- Các câu có phủ định ('not') hoặc mệnh đề phụ ('or', 'but not through ...') thường khiến mô hình TF-IDF và W2V trung bình dự đo
- LSTM (đặc biệt bản pre-trained) có xu hướng hiểu ngữ cảnh tốt hơn, vì trạng thái ẩn theo chuỗi giúp mô hình nhận diện được mối
- Nếu LSTM (pre-trained) chính xác hơn bản học từ đầu, điều đó chứng tỏ embedding Word2Vec cung cấp ngữ nghĩa ban đầu hữu ích, g
```