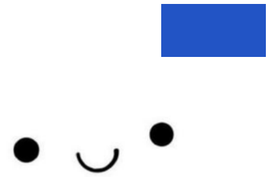
Intent Classification

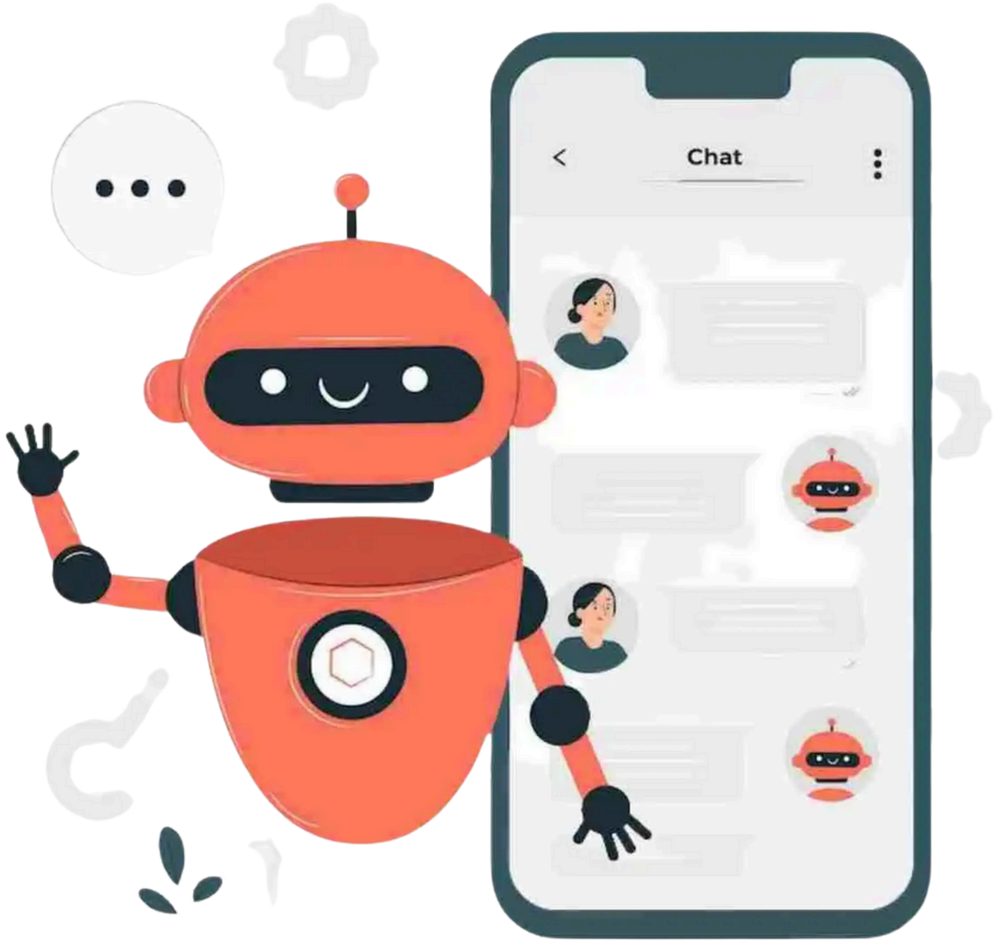


Hoàng Hồng Quân
HE195052

Nguyễn Hải Anh
HE172327

Nguyễn Minh Giang
HE195096

Hồ Viết Hoàng
HE 205149

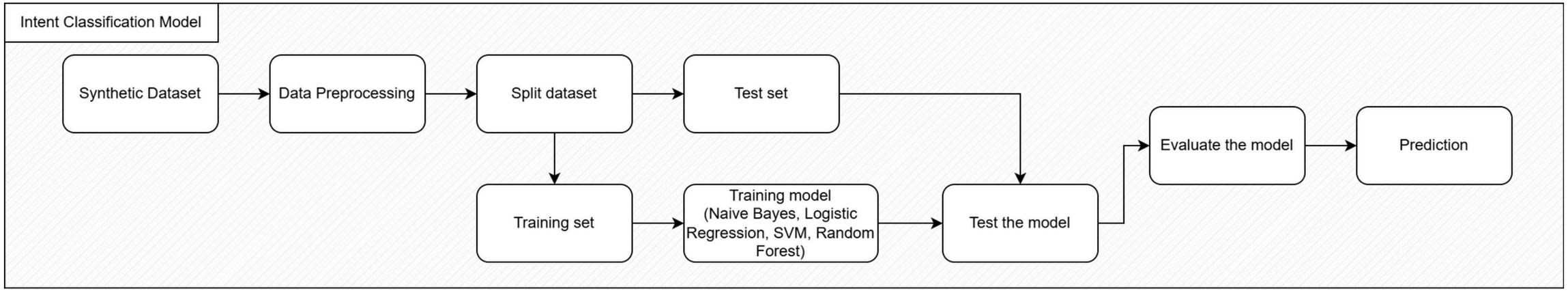


Giới thiệu dự án

Hiện nay, nhu cầu xây dựng chatbot ngày càng tăng nhờ sự phát triển của các mô hình ngôn ngữ lớn. Tuy nhiên, việc áp dụng còn gặp phải các rào cản về chi phí, độ trễ và đặc biệt là sự thiếu hụt dữ liệu chuyên ngành.

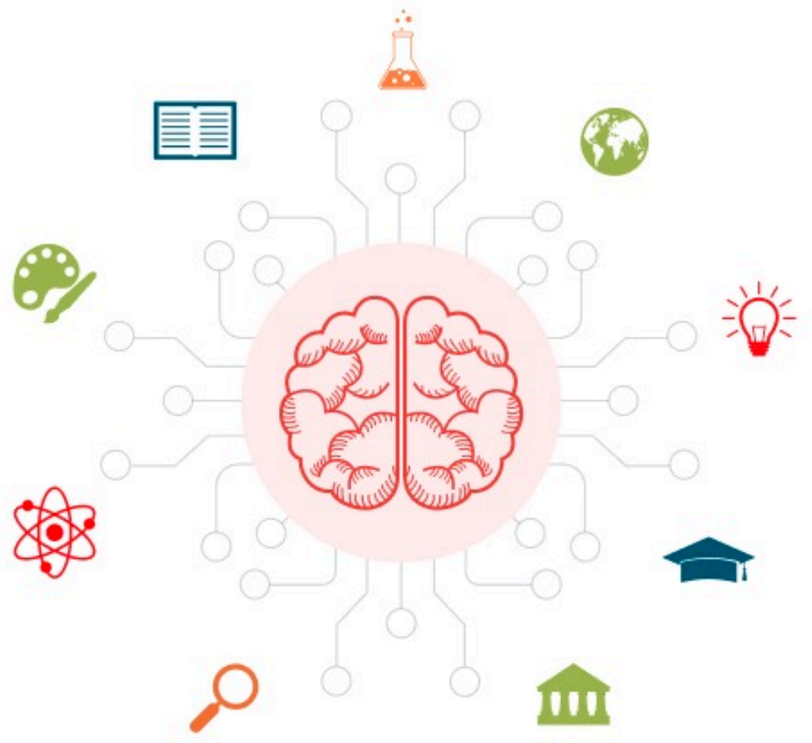
Để một chatbot thực sự hiệu quả, nó phải có khả năng phân loại chính xác ý định (intent) đằng sau mỗi câu hỏi của người dùng. Do đó, dự án này tập trung vào bài toán cốt lõi là xây dựng mô hình Intent Classification bằng cách sử dụng synthetic data.

Pipeline


Intent Classification Model
Synthetic Dataset
Data Preprocessing
Split dataset
Test set
Training set
Training model
(Naive Bayes, Logistic
Regression, SVM, Random
Forest)
Test the model
Evaluate the model
Prediction

Synthetic data

- lich_hoc: Các câu hỏi liên quan đến thời khóa biểu hàng tuần (thời gian, địa điểm, giảng viên).
- lich_thi: Các câu hỏi về lịch thi (ngày thi, phòng thi, hình thức thi).
- diem_danh: Các câu hỏi về tình trạng chuyên cần, số buổi vắng mặt.
- diem_so: Các câu hỏi về điểm thi, điểm thành phần, điểm tổng kết và GPA.
- hoc_phi: Các câu hỏi về học phí môn học.



Data Preprocessing

```
def normalize_unicode(text):
    return unicodedata.normalize('NFC', text)

def remove_accents(text):
    for pattern, repl in patterns.items():
        text = re.sub(pattern, repl, text)
    return text
```

Dùng để chuẩn hóa mã Unicode tiếng Việt, đảm bảo các ký tự có dấu được biểu diễn thống nhất.
Dùng regex trong bảng patterns ở trên để xóa dấu tiếng Việt

Feature extraction

Vectorize features:

- Giúp mô hình hiểu được nội dung văn bản bằng cách gán trọng số cho từ quan trọng.

```
vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(X_train_text)
X_test = vectorizer.transform(X_test_text)
```

Word-embedding features

```
embedding_size = 100
token_sequences = df['tokens'].tolist()

w2v_model = Word2Vec(
    sentences=token_sequences,
    vector_size=embedding_size,
    window=5,
    min_count=1,
    workers=4,
    sg=1,
    epochs=200,
    seed=42
)

def sentence_vector(tokens, model=w2v_model, dimension=embedding_size):
    valid_tokens = [token for token in tokens if token in model.wv.key_to_index]
    if not valid_tokens:
        return np.zeros(dimension)
    return np.mean(model.wv[valid_tokens], axis=0)

X_train_embed = np.vstack([sentence_vector(tokens) for tokens in X_train_tokens])
X_test_embed = np.vstack([sentence_vector(tokens) for tokens in X_test_tokens])
```

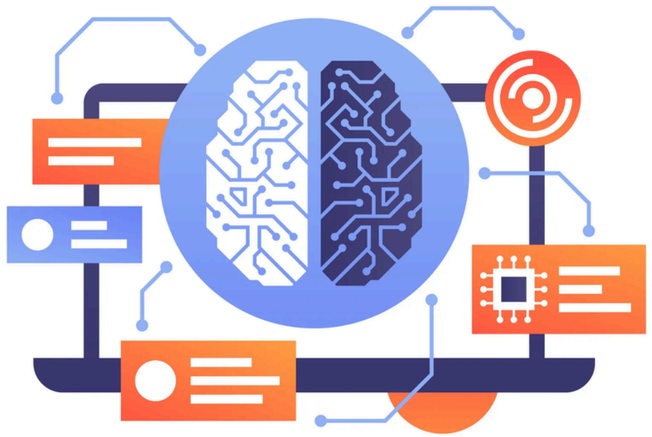
Define Models

- Multinomial Naive Bayes
- Logistic Regression
- Support Vector Machine
- Random Forest

```
models = {
    "Multinomial Naive Bayes": MultinomialNB(),
    "Logistic Regression": LogisticRegression(random_state=42),
    "Support Vector Machine": SVC(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
}
```

- Baseline Models

```
baseline_models = {
    "Most Frequent Baseline": DummyClassifier(strategy='most_frequent'),
    "Stratified Baseline": DummyClassifier(strategy='stratified', random_state=42)
}
```



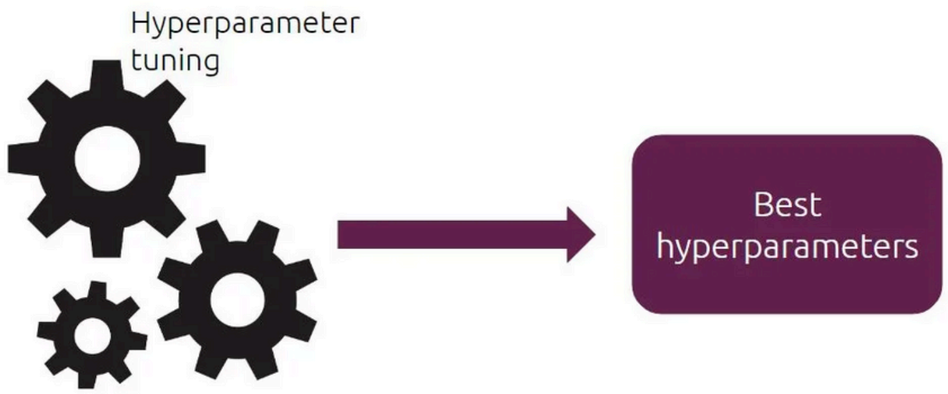
Train Models

- Huấn luyện các mô hình phân loại ý định trên dữ liệu đã xử lý.
- Tối ưu hyperparameters cho những mô hình cần tinh chỉnh.
- Đánh giá hiệu suất mô hình bằng tập kiểm tra (test set).
- So sánh kết quả giữa các mô hình để chọn mô hình tốt nhất phục vụ chatbot.

Training

Training Process & Hyperparameter Optimization

- Tiến hành huấn luyện từng mô hình trên tập train sử dụng đặc trưng TF-IDF.
- Với các mô hình có tham số cần tinh chỉnh → áp dụng GridSearchCV.
- Cross-validation giúp mô hình tổng quát tốt hơn, hạn chế overfitting.
- Với mô hình không cần tinh chỉnh → huấn luyện trực tiếp.



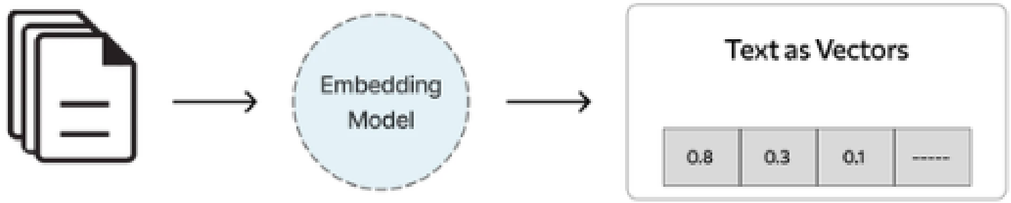
```python
param_grids = {
    "Logistic Regression": {
        "C": np.logspace(-4,4,20),
        "solver": ["liblinear"],
    },
    "Support Vector Machine": {
        "C": [1, 5, 10]
    },
    "Random Forest": {
        "n_estimators": [100, 200, 300, 400, 500]
    }
}

for model_name, model in models.items():
    if model_name in param_grids:
        clf = GridSearchCV(model, param_grid = param_grids[model_name], cv = 3, verbose=False, n_jobs=-1)
        best_clf = clf.fit(X_train, y_train)
        model = best_clf.best_estimator_
        print("Best hyperparameters of", model_name, "is:", end=" ")
        try:
            print(model.best_params_)
        except:
            print(model)
    else:
        continue
    y_pred = model.predict(X_test)
```

```
Best hyperparameters of Logistic Regression is: LogisticRegression(C=np.float64(11.288378916846883), random_state=42,
                   solver='liblinear')
Best hyperparameters of Support Vector Machine is: SVC(C=5, random_state=42)
Best hyperparameters of Random Forest is: RandomForestClassifier(n_estimators=400, random_state=42)
```

Embedding models

- Huấn luyện các mô hình Embedding.

- Với 3 mô hình:
  - Logistic Regression
  - Support Vector Machine
  - Random Forest

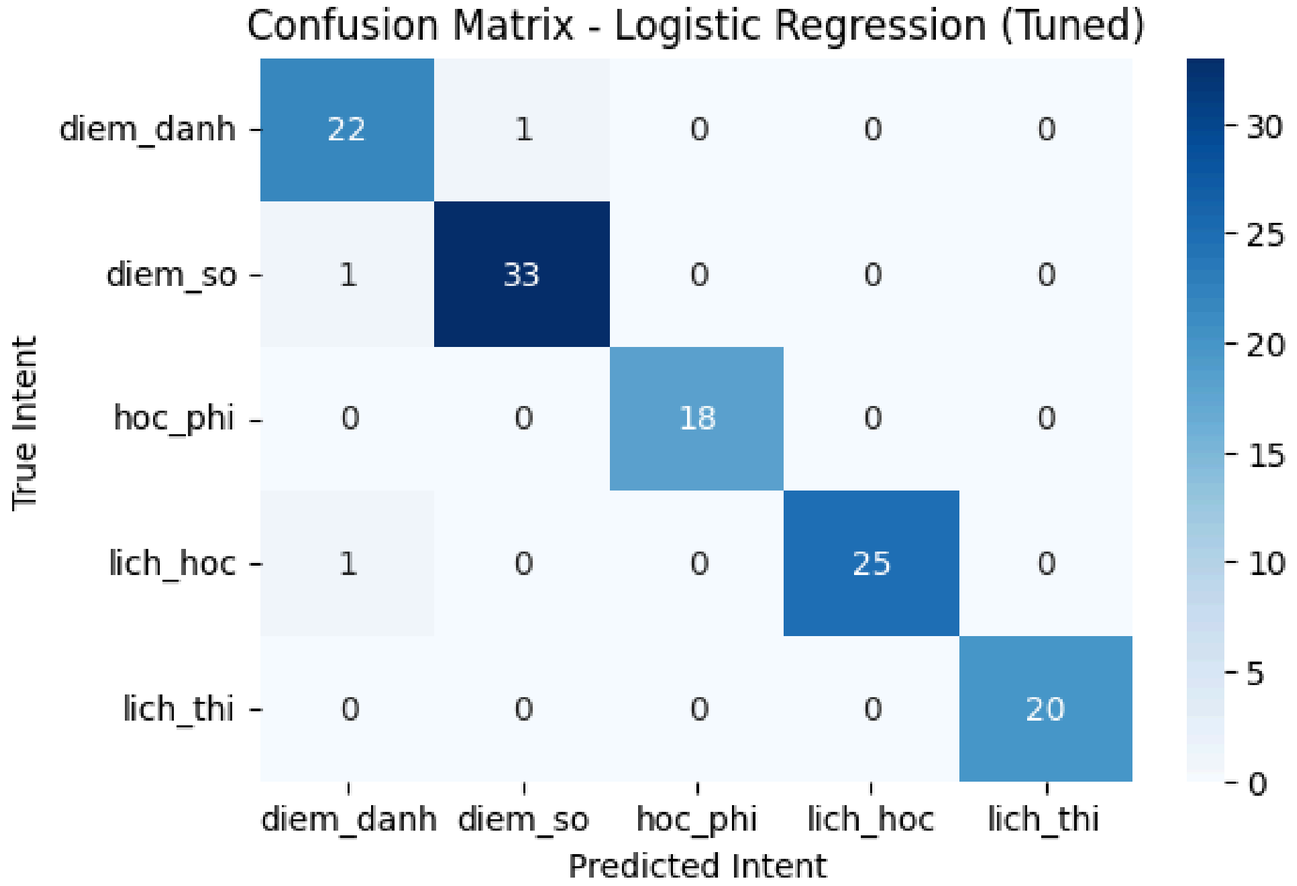



```
embedding_models = {
    "Logistic Regression (Embeddings)": LogisticRegression(max_iter=1000, random_state=42),
    "Support Vector Machine (Embeddings)": SVC(probability=True, random_state=42),
    "Random Forest (Embeddings)": RandomForestClassifier(random_state=42)
}
```

Result

	Category	Model	Precision	Precision Tuned	Recall	Recall Tuned	F1-Score	F1-Score Tuned	Accuracy	Accuracy Tuned
0	Baseline	Most Frequent Baseline	0.079	None	0.281	None	0.123	None	0.281	None
1	Baseline	Stratified Baseline	0.192	None	0.182	None	0.185	None	0.182	None
2	Trained Model	Multinomial Naive Bayes	0.922	None	0.917	None	0.916	None	0.917	None
3	Trained Model	Logistic Regression	0.953	0.976 Δ+0.023	0.950	0.975 Δ+0.025	0.951	0.975 Δ+0.024	0.950	0.975 Δ+0.025
4	Trained Model	Support Vector Machine	0.960	0.969 Δ+0.009	0.959	0.967 Δ+0.008	0.959	0.967 Δ+0.008	0.959	0.967 Δ+0.008
5	Trained Model	Random Forest	0.944	0.952 Δ+0.008	0.942	0.95 Δ+0.008	0.942	0.95 Δ+0.008	0.942	0.95 Δ+0.008
6	Embedding Model	Logistic Regression (Embeddings)	0.967	None	0.967	None	0.967	None	0.967	None
7	Embedding Model	Support Vector Machine (Embeddings)	0.967	None	0.967	None	0.967	None	0.967	None
8	Embedding Model	Random Forest (Embeddings)	0.945	None	0.942	None	0.942	None	0.942	None

Confusion Matrix



01

Dataset sử dụng

- Nguồn dữ liệu: File intents.csv.
- Tổng số mẫu: 602 câu.
- Gồm 5 intent:
1. diem_danh
2. diem_so
3. hoc_phi
4. lich_hoc
5. lich_thi
- Tiền xử lý:
1. Chuyển chữ thường.
2. Chuẩn hóa Unicode.
3. Xóa dấu.
4. Tách từ.

02

FEATURES

- TF-IDF:
1. Sử dụng TfidfVectorizer để chuyển đổi văn bản thành vector dựa trên tần suất và độ hiếm của từ.
- Word Embeddings (Word2Vec):
1. Sử dụng Gensim Word2Vec (vector_size=100) để học ngữ nghĩa của từ.
2. Vector của câu được tính bằng cách lấy trung bình (average) vector của các từ trong câu.

03

Mô hình

- Baseline Models:
1. DummyClassifier (Most Frequent): Accuracy: 28.1%
2. DummyClassifier (Stratified): Accuracy: 18.2%
- Main Models:
1. Multinomial Naive Bayes.
2. Logistic Regression
3. Support Vector Machine (SVM)
4. Random Forest

04

TINH CHỈNH & KẾT QUẢ

- Tuning:
1. Dùng GridSearchCV trên các mô hình để tìm hyperparameter tốt nhất.
2. Các tham số được tune chính: C (cho Logistic Regression & SVM), Sô cây (cho Random Forest).
- Kết quả:
1. Mô hình Tốt nhất: Logistic (Accuracy: 97.5%)

Thank you