AIB LAB

V J

May 19, 2025

0.1 1. Marketer to Machine: Develop a ML model for Smart email Compose. Smart email compose finishes sentences for you by predicting what word or words user will type next.

[1]:
```
import random
from collections import defaultdict

def train_model(sentences):
    word_dict=defaultdict(list)
    for sentence in sentences:
        words=sentence.split()
        for i in range(len(words)-1):
            word_dict[words[i]].append(words[i+1])
    return word_dict
def predict_next_word(word_dict,text):
    words=text.split()
    last_word=words[-1] if words else ""
    return random.choice(word_dict.get(last_word,["..."]))
sentences=[
    "Hello how are you",
    "I am doing great",
    "Nice to meet you",
    "Hope you are Having a good day",
    "looking forword to our meeting"
]


word_dict = train_model(sentences)

#get user input and predidct next word
user_input = input("Type a sentence")
next_word = predict_next_word(word_dict, user_input)
print(f"Predict next word: {next_word}")
```

```
Type a sentence Hope
```

```
Predict next word: you
```

0.2 2. Marketer to Machine: Develop level-2 Marketer-to-Machine (M2M) Scale of intelligent automation to

personalize business email based on user preferences and interests (Extension to Exp. No.1).

[2]:
```
import random
def generate_email(user_name,user_intrest,company_name,product):
    greetings=[
        f"Hi {user_name},",
        f"Hello {user_name},",
        f"Dear {user_name},",
    ]
    opening_lines=[
    f"Hope your doing well as someone intrested in {user_intrest}, we have␣
 ↪smething exiting for you.",
    f"I wanted to persomally reach out since we know your passion for␣
 ↪{user_intrest}.",
    f"Since your intrested in {user_intrest},we thought you'd love to here about␣
 ↪this."
    ]

    body =f"At {company_name} , we are excited to introduce our  latest␣
 ↪{product} , designed to enhance you experience with {user_intrest}"

    closing_lines = [
    "Let's schedule a quick chat to explore how this can benefit you",
    "would love to hear your thoughts , Let's connect soon!",
    "Looking forward to your feedback, Let's discuss further"
    ]

    signature = f"Best Regards ,\n The {company_name} Team",

    email =f"{random.choice(greetings)}\n\n{random.
 ↪choice(opening_lines)}\n\n{body}\n\n{random.
 ↪choice(closing_lines)}\n\n{signature}"

    return email


user_name ="Name"
user_intrest = "Machine Learning"
company_name = "CEC"
product = "PC"
print(generate_email(user_name,user_intrest,company_name,product))
```

```
Dear Name,

I wanted to persomally reach out since we know your passion for Machine
```

```
Learning.

At CEC , we are excited to introduce our  latest PC , designed to enhance you
experience with Machine Learning

would love to hear your thoughts , Let's connect soon!

('Best Regards ,\n The CEC Team',)
```

0.3 3. AI and Marketing: Develop data-driven content for a given business organization (Web site). Optimize Website content for search engines. Send emails to customers with personalized content/activity.

[3]:
```python
from flask import Flask, request, render_template_string

app = Flask(__name__)

# Sample customer data
customers = [
    {"name": "Alice", "email": "alice@example.com", "interest": "Science␣
 ↪Fiction"},
    {"name": "Bob", "email": "bob@example.com", "interest": "Mystery"},
    {"name": "Carol", "email": "carol@example.com", "interest": "Romance"}
]

# SEO Content Generator
def generate_seo(product, keywords):
    return f"""
    <h1>Buy {product} Online</h1>
    <p>Looking for {product}? Discover {', '.join(keywords)} and more!</p>
    <p>Tags: {', '.join(keywords)}</p>
    """

# Email Personalizer
def personalize_emails(customers):
    emails = []
    for c in customers:
        subject = f"Hi {c['name']}, check out new {c['interest']} books!"
        body = f"""
Dear {c['name']},

You love {c['interest']} books, and we've got new ones just for you!

Cheers,
Bookstore Team
"""
        emails.append({'email': c['email'], 'subject': subject, 'body': body})
```

```
    return emails

# HTML Template
html_template = """
<!DOCTYPE html>
<html>
<head>
    <title>AI Marketing Tool</title>
</head>
<body>
    <h2>SEO Content Generator</h2>
    <form method="POST">
        <label>Product:</label><br>
        <input name="product" type="text" required><br><br>

        <label>Keywords (comma-separated):</label><br>
        <input name="keywords" type="text" required><br><br>

        <input type="submit" value="Generate SEO & Emails">
    </form>

    <hr>

    {% if seo_result %}
        <h3>Generated SEO Content</h3>
        <div style="border:1px solid #ccc; padding:10px;">{{ seo_result|safe }}</
 ↪div>
    {% endif %}

    <h3>Personalized Emails</h3>
    {% for email in emails %}
        <div style="border:1px solid #ccc; padding:10px; margin-bottom:10px;">
            <strong>To:</strong> {{ email.email }}<br>
            <strong>Subject:</strong> {{ email.subject }}<br>
            <pre>{{ email.body }}</pre>
        </div>
    {% endfor %}
</body>
</html>
"""

# Main route
@app.route("/", methods=["GET", "POST"])
def index():
    seo_result = ""
    if request.method == "POST":
        product = request.form.get("product", "").strip()
```

```
        keywords_raw = request.form.get("keywords", "")
        keywords = [k.strip() for k in keywords_raw.split(",") if k.strip()]
        if product and keywords:
            seo_result = generate_seo(product, keywords)
    emails = personalize_emails(customers)
    return render_template_string(html_template, seo_result=seo_result,␣
↪emails=emails)

# Entry point
if __name__ == "__main__":
    app.run(port=5000, debug=True ,use_reloader=False)
```

```
 * Serving Flask app '__main__'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

0.4 4.AI and Marketing: Develop a system to recommend highly targeted content to users of the Web site

(Extension to Exp. No 3).

[6]:
```
from flask import Flask, request, render_template_string
import csv

app = Flask(__name__)

def load_books():
    with open("books.csv", newline='') as f:
        return list(csv.DictReader(f))

def recommend_books(user_interest, books):
    return [b for b in books if b["genre"].lower() == user_interest.lower()]

html_template = """
<!DOCTYPE html>
<html>
<head><title>AI Book Recommender</title></head>
<body>
<h2>Smart Book Recommender</h2>
<form method="POST">
Name: <input name='name'><br><br>
Interest (e.g., Romance, Mystery): <input name='interest'><br><br>
<input type='submit' value='Recommend'>
</jorm>
```

```
{% if recommendations %}
<h3>Hi {{ name }}, here are your book suggestions:</h3>
<ul>
{% for r in recommendations %}
<li><strong>{{ r.title }}</strong>: {{ r.description }}</li>
{% endfor %}
</ul>
{% endif %}
</body>
</html>
"""

@app.route("/", methods=["GET", "POST"])
def index():
    recommendations = []
    user_name = ""
    user_interest = ""
    if request.method == "POST":
        user_name = request.form["name"]
        user_interest = request.form["interest"]
        books = load_books()
        recommendations = recommend_books(user_interest, books)
    return render_template_string(html_template, name=user_name,
                                  interest=user_interest,␣
 ↪recommendations=recommendations)

if __name__ == "__main__":
    app.run(port=5001,debug=False, use_reloader=False)
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
 * Running on http://127.0.0.1:5001
Press CTRL+C to quit
```

0.5 5. AI and Advertisement: Develop AI-powered programmatic advertising by using cookies and collecting user data.

[7]:
```
from sklearn.ensemble import RandomForestClassifier
import numpy as np

training_data = np.array([
    [10 ,1],[14, 2],[18, 1],[20, 2],[12, 1],
])
labels = np.array([1, 0, 1, 0, 1])
```

```
#train the model
model = RandomForestClassifier()
model.fit(training_data,labels)

ads_pool = [
    {"ad_text":"Buy Shoes - 50% off!","tag":1},
    {"ad_text":"try new cofee","tag":2},
    {"ad_text":"EAT and PAY","tag":3}
]

def get_ad(hour: int , location: str):
    loc = location.lower()
    if loc == "india":
        loc_code = 1
    elif loc == "usa":
        loc_code = 2
    else:
        loc_code = 3

    features = [hour, loc_code]
    click_prob= model.predict_proba([features])[0][1]

    if click_prob > 0.7:
        return ads_pool[0]
    elif click_prob >0.4 :
        return ads_pool[1]
    else:
        return ads_pool[2]


hour  = int(input("Enter hour :"))
location =input ("Enter the location")
recommended_ad = get_ad(hour , location)
print(f"Recommended Ad for {location} at {hour}:", recommended_ad["ad_text"])
```

```
Enter hour : 5
Enter the location India

Recommended Ad for India at 5: Buy Shoes - 50% off!
```

[ ]: