Introduction

Navigating today's job market can feel like trying to find your way through a maze – it's often tough to pinpoint a career path that truly fits your unique interests and strengths. But what if there was a smarter way to get personalized guidance? That's where Artificial Intelligence (AI) comes in. We're proposing an AI-powered virtual career counselor designed to transform how people discover their ideal professions. This intelligent system uses Natural Language Processing (NLP) to have natural conversations with you, understand what you're looking for, and then suggest career options that are just right for you. Think of it as having a personal career coach, available whenever you need them.

Abstract

This project focuses on designing and implementing an AI-based virtual career counselling system using machine learning and NLP techniques. The system is built using Python, NLTK, and the Rasa framework for natural language understanding and conversation management. Streamlit is used for creating an interactive web interface. Users can input their interests, and the chatbot will process this data and recommend potential career paths. The goal is to provide a supportive and intelligent platform for users seeking career guidance based on their expressed interests and preferences.

Objective

Build an NLP chatbot that recommends career paths based on user interests and input.

TOOLS & TECHNOLOGIES:

- Python
- NLTK
- Rasa (for chatbot framework)
- Streamlit (for UI)
-

Steps involved in developing the project:

Step 1. Define Intents

- **Create a list of possible intents such as:**
    - tech_interest (e.g.,"I love programming")
    - arts_interest(e.g.,"I'm interested in painting")
    - commerce_interest(e.g., "I like managing money")
    - career_query(e.g., "What career suits me?")

Step 2. Preprocessing with NLTK

- **Clean user inputs by:**
    - Tokenizing
    - Removing stop words
    - Lemmatization

```
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

def preprocess(text):
    tokens = word_tokenize(text.lower())
    filtered = [lemmatizer.lemmatize(w) for w in tokens
if w.isalpha() and w not in stop_words]
    return filtered
```

Step 3. Rasa Chatbot Setup

- Install Rasa: pip install rasa
- Create Rasa project: rasa init
- Define nlu.yml with intents and training phrases
- Update domain.yml with intents and responses
- Define stories in stories.yml
- Add rules for chatbot flow
- Train model: rasa train
- Test: rasa shell

Step 4. Career Logic (Custom Action)

- Add logic in actions.py:

```
from rasa_sdk import Action
from rasa_sdk.executor import CollectingDispatcher

class ActionRecommendCareer(Action):
    def name(self):
        return "action_recommend_career"

    def run(self, dispatcher, tracker, domain):
        user_message = tracker.latest_message.get("text")
        if "code" in user_message or "programming" in
user_message:
```

```
        career = "Software Developer, Data Scientist"
    elif "paint" in user_message or "art" in user_message:
        career = "Graphic Designer, Animator"
    elif "finance" in user_message or "business" in user_message:
        career = "Chartered Accountant, Business Analyst"
    else:
        career = "Career Counselor needed for more info."
        dispatcher.utter_message(text=f"Based on your interest, you can explore: {career}")
    return
```

Step 5. Streamlit Frontend

```
import streamlit as st
import requests

st.title("AI Career Counsellor")
user_input = st.text_input("Tell me about your interests:")
if user_input:
    response = requests.post("http://localhost:5005/webhooks/rest/webhook", json={"message": user_input})
    for r in response.json():
        st.write(": ", r["text"])
```

Step 6. Deployment

- Host chatbot using Rasa X or Docker
- Deploy Streamlit UI using Streamlit Cloud

Python Code :

```
career_matches = [
    ("Data Scientist", 0.92),
    ("Financial Analyst", 0.85),
    ("Product Manager", 0.73),
    ("Marketing Specialist", 0.32)
]

career_knowledge_base = {
    "Data Scientist": {
        "description": "Data Scientists extract insights from structured and unstructured data using statistical and machine learning techniques.",
        "education": "B.Tech in Data Science or related field, or an M.Sc in Statistics/AI",
        "responsibilities": [
            "Building predictive models and machine learning algorithms",
            "Analyzing large volumes of data for actionable insights",
            "Collaborating with cross-functional teams to solve business problems"
        ],
        "industries": ["technology", "finance", "healthcare"]
    },
    "Financial Analyst": {
        "description": "Financial Analysts evaluate financial data to help organizations make investment decisions.",
        "education": "B.Com, BBA, or MBA in Finance",
        "responsibilities": [
            "Conducting financial forecasting, reporting, and operational metrics tracking",
            "Evaluating capital expenditures and asset depreciation",
            "Modeling financial data to support decision-making"
        ],
        "industries": ["finance", "banking", "corporate"]
    },
    "Product Manager": {
        "description": "Product Managers oversee the development and delivery of digital or physical products.",
        "education": "Bachelor's degree in Business, Engineering, or related field",
        "responsibilities": [
            "Defining product vision and roadmap",
            "Collaborating with engineering and design teams",
            "Tracking product performance metrics"
        ],
        "industries": ["tech", "consumer goods", "startups"]
    },
    "Marketing Specialist": {
        "description": "Marketing Specialists craft and execute campaigns to promote products or services.",
        "education": "Bachelors in Marketing, Mass Communication, or Business",
        "responsibilities": [
            "Creating marketing content and assets",
            "Monitoring campaign performance",
            "Conducting market research and analysis"
        ],
        "industries": ["advertising", "e-commerce", "retail"]
    }
}

def generate_recommendation(career_scores, user_experience, user_interests, user_skills, user_values, career_knowledge_base):
    """
    Generate a formatted recommendation string with detailed career information and next steps.

    Args:
        career_scores (list): A list of tuples (career_path, score), sorted by score.
        user_experience (str): A description of user's work experience.
        user_interests (list): A list of user's interests.
        user_skills (list): A list of user's skills.
        user_values (list): A list of user's work values.
        career_knowledge_base (dict): The knowledge base of career paths.

    Returns:
        str: A string containing detailed career recommendations and next steps.
    """
    result = "Based on your profile and experience, here are some potential career paths for you:\n\n"
    has_recommendations = False

    for career, score in career_scores:
```

```
    if score > 0:  # Only consider careers with a positive match score
        has_recommendations = True
        career_info = career_knowledge_base.get(career, {})

        result += f"### {career} (Match Score: {score:.2f})\n"
        result += f"**Description:** {career_info.get('description', 'N/A')}\n"
        result += f"**Relevant Education:** {career_info.get('education', 'N/A')}\n"
        result += "**Typical Responsibilities:**\n"
        responsibilities = career_info.get('responsibilities', [])
        if responsibilities:
            for resp in responsibilities:
                result += f"- {resp}\n"
        else:
            result += "- N/A\n"

        # Analyze experience for relevance
        experience_keywords = user_experience.lower().split()
        relevant_keywords = []
        description = career_info.get("description", "")
        industries = career_info.get("industries", [])


        for keyword in experience_keywords:
            if keyword in description.lower() or any(keyword in resp.lower() for resp in responsibilities) or any(keyword in ind.lower() for ind in industries):
                relevant_keywords.append(keyword)

        if relevant_keywords:
            result += f"**Relevance of your experience:** Your experience with {' and '.join(set(relevant_keywords))} seems relevant to this role.\n"

        # Suggest next steps
        result += "**Potential Next Steps:**\n"
        result += "- Research this career path further online.\n"
        if career_info.get('education') != 'N/A':
            result += f"- Look into relevant educational programs like {career_info.get('education', 'a relevant degree')}.\n"
        result += "- Consider informational interviews with professionals in this field.\n"
        result += "- Explore online courses or certifications to build relevant skills.\n"
        result += "\n" # Add a newline for spacing between recommendations


    if not has_recommendations:
        result = ("Your unique combination of interests, skills, and experience is inspiring! "
                  "However, based on the current knowledge base, I recommend exploring online career assessment tools or speaking to a career mentor "
                  "for deeper insights.")

    result += f"**Your Profile Summary:**\n"
    result += f"- Interests: {', '.join(user_interests)}\n"
    result += f"- Skills: {', '.join(user_skills)}\n"
    result += f"- Values: {', '.join(user_values)}\n"
    result += f"- Past Experience: {user_experience}\n"


    return result

# Example usage with previously calculated career_matches and user profile
user_interests = ["technology", "finance"]
user_skills = ["programming", "data analysis", "problem-solving"]
user_values = ["innovation", "learning", "autonomy"]
user_experience = "Worked as a financial analyst for 3 years."


career_advice = generate_recommendation(
    career_matches,
    user_experience,
    user_interests,
    user_skills,
    user_values,
    career_knowledge_base
)

print(career_advice)
```

Deliverables

- Rasa project folder (including training data, domain, actions)
- Streamlit frontend application
- Sample interaction video or demo recording

Conclusion

Ultimately, our **AI Virtual Career Counselor** offers a fresh, innovative solution to help you explore your career possibilities with intelligent automation. By thoughtfully analyzing your input and connecting it with relevant career paths, this chatbot makes personalized career advice more accessible than ever before. With future enhancements like understanding your emotions through **sentiment analysis** or integrating with up-to-date **real-world job market data**, this system has the potential to become an incredibly powerful tool for anyone seeking career guidance – from students just starting out to experienced professionals and even educational institutions.