REVOLUTIONIZING MENTORSHIP WITH ML

-Avishi, Shubham and Siddhant

PROBLEM STATEMENT

Developing a mentor recommender system that suggests relevant professors to students based on their queries, such as interests in specific areas (e.g., design).

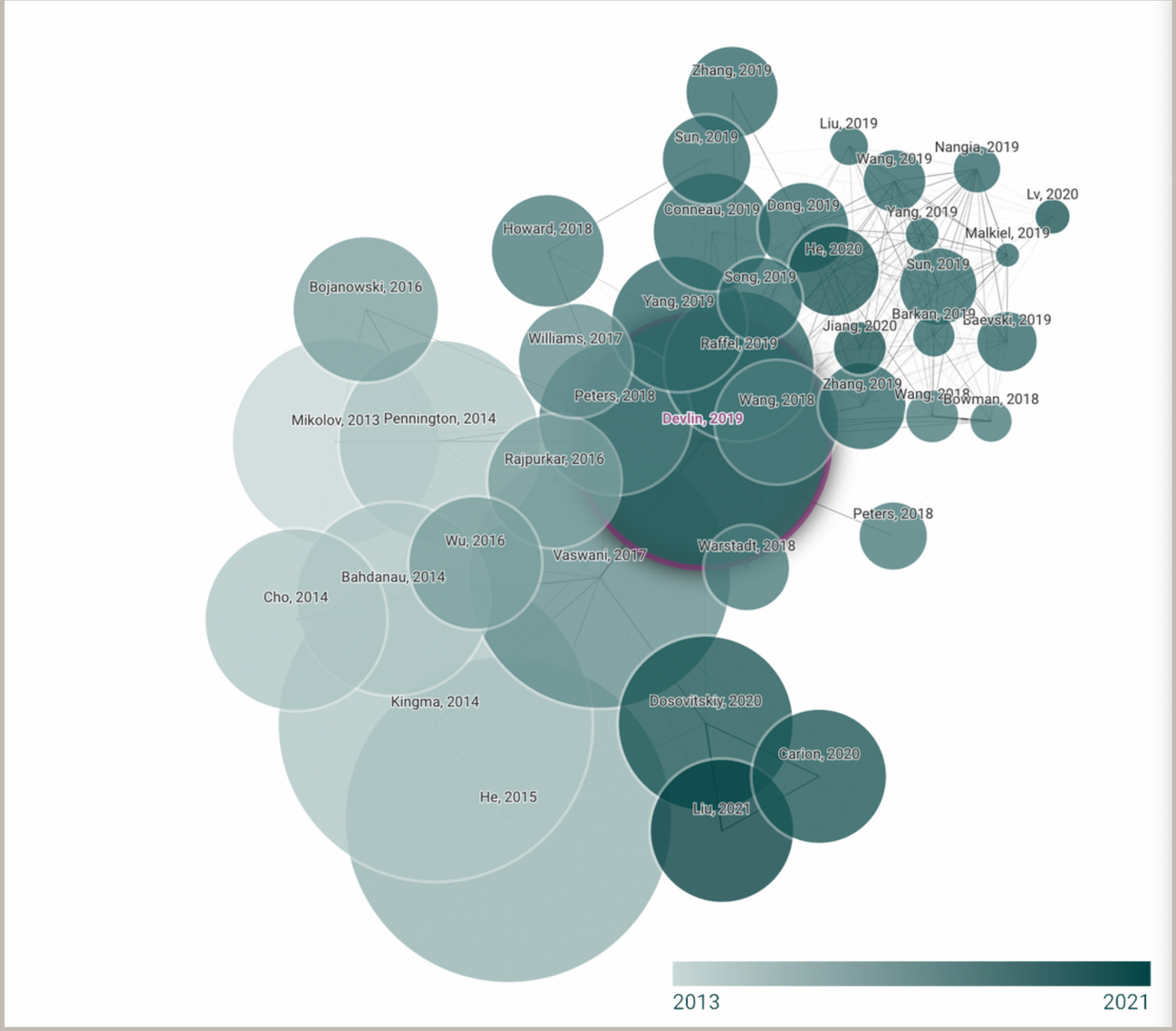
Potential Applications

- Academic institutions: In schools and colleges, students can be matched to their mentor soulmates
- Professional Development: getting career advice from someone who's been there, done that.
- Skill Development Platforms: this can also be extended to online skill development platforms to recommend the courses which perfectly align with your aim.

Potential Impact

- Time and Resource efficiency
- Optimized matching
- Improved learning outcomes

LITERATURE REVIEW



(Source: Connected Papers)

RELATED WORK IN NLP AND TEXT PROCESSING:

‘BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding’ by Devlin et al.:

We delved into this paper to understand BERT's architecture and pre training process. BERT's pre-trained embeddings will play a crucial role in understanding the context of professor descriptions in our recommender system.

PREVIOUS RESEARCH ON MENTOR RECOMMENDER SYSTEMS:

‘A Survey of Recommender Systems’ by Adomavicius and Tuzhilin:

This survey paper provides an overview of various recommendation systems, including collaborative filtering and content-based filtering. It helped us understand the foundational concepts of recommendation systems and their applications.

DATASET

Data Collection Method: We made a web scraper to collect data from academic sources such as university websites, and official faculty profiles.

Ethical Considerations: No ethical concerns were encountered as the data collected was publicly available and did not contain personal or sensitive information.

Dataset Size: The dataset comprises information on a substantial number of professors, which includes their names, areas of expertise, research projects, and small descriptions. (217)

Missing Data Handling: Employing strategies that involve initial data collection and the exploration of additional data sources.

SCRAPER FUNCTION

```
driver = webdriver.Chrome()

def scrape_all_pages(base_url, start_page, num_pages):
    all_results = pd.Series()

    for page_number in range(start_page, start_page + num_pages):
        url = f"{base_url}/page/{page_number}/"

        # Navigate to the page
        driver.get(url)

        # Find the elements that contain the links to professor's pages
        elements = driver.find_elements(By.XPATH, "/html/body/section[4]/div/div/div/div/div/a")

        # Extract URLs from the elements
        professor_urls = [element.get_attribute('href') for element in elements]
        professor_urls = pd.Series(professor_urls)
        all_results = pd.concat([all_results, professor_urls], axis = 0)
    return all_results

# Usage
base_url = "https://www.ashoka.edu.in/roles/faculty"
num_pages = 17  # Set the number of pages you want to scrape

urls = scrape_all_pages(base_url, 2, num_pages)

def get_bio(url):
    driver.get(url)

    # Extracting the name
    name = driver.find_element(By.XPATH, '/html/body/section[2]/div[1]/div/div/div/div/h2').text

    # Extracting the bio
    bio_containers = driver.find_elements(By.XPATH, '/html/body/section[3]/div/div/div/div[2]/div[1]/div/div/div')
    bio_texts = []

    for container in bio_containers:
        paragraphs = container.find_elements(By.TAG_NAME, 'p')
        for paragraph in paragraphs:
            bio_texts.append(paragraph.text)

    # Concatenate paragraphs to form a single bio text
    bio = ' '.join(bio_texts)

    return {'name': name, 'bio': bio}
```

GMM

Step 1: Create a TF-IDF vectorizer for the tags column and generate the probability matrix.

Step 2: Identify the number of clusters (5)

Step 3: Apply GMM and form clusters

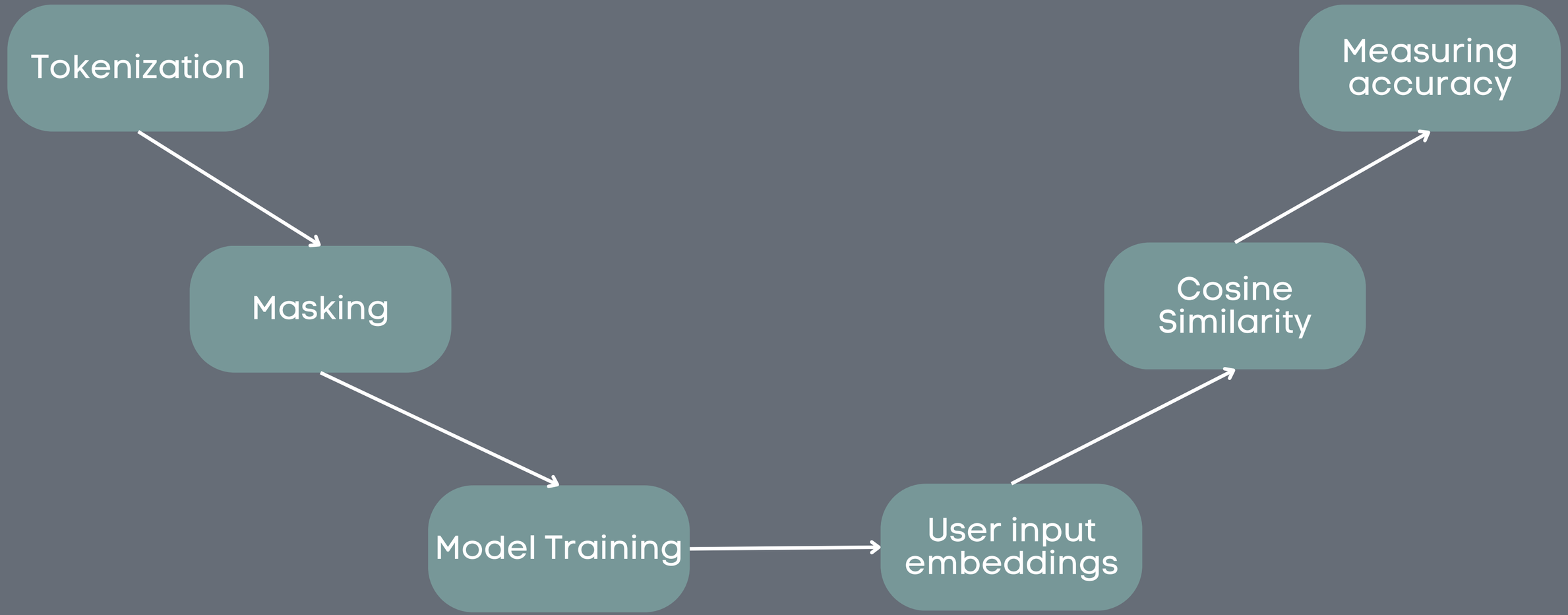
Step 4: Accept the user preference, vectorize it and map it to the professors

```
# Example usage
new_user_input = "biology"
recommendations = recommend_based_on_input(new_user_input)
print(recommendations)
```

✓ 0.0s

```
13          Dr. Subhasis Ray
24          Dr. Manoj Kannan
25     Dr. Sandeep Manjanna
26        Dr. Monika Sharma
28       Dr. Nandini Kannan
```

ML METHODOLOGY



FEATURE PREPROCESSING: TOKENIZER

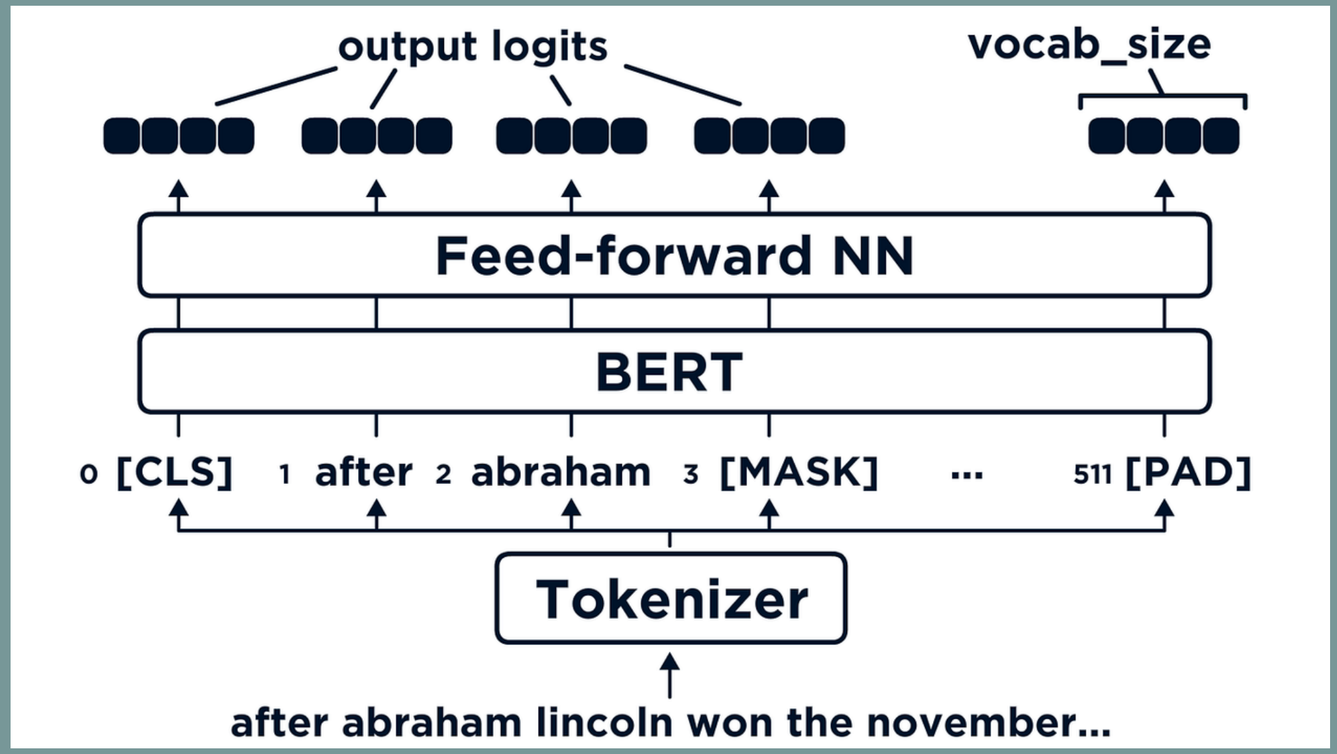
Adapting raw text to a format suitable for the BERT model:

Text converted into BERT-compatible format

- **Tokenization**
- **Padding**: Sequences shorter than the maximum length are padded with a special [PAD] token to match the maximum length
- **Truncation**: BERT has a maximum input length. Sequences longer than this length are truncated (512)
- Transformation into **tensor format**: Necessary for compatibility with PyTorch

MASKED LANGUAGE MODELLING

- BERT considers context from both directions
- Some tokens in the input are randomly selected and replaced with the [MASK] token.
- The model is then trained to predict the original words based on the context of the surrounding words, facilitating the model's ability to predict and understand context.



USER INPUT

- Use our pre-trained model to create BERT embeddings of the input
- Mean pooling to capture the overall semantic meaning

MATCHING

- Calculate cosine-similarity scores between the input embeddings and the faculty bios
- Recommend the top 5 based on these scores

EXAMPLE

In [17]:
```
user_input = 'I am interested in the advancements in robotic surgery and nanotechnology, in the field of neuroscience and microbi
# Tokenize and obtain BERT embeddings for the user input
tokenized_user = tokenizer(user_input, return_tensors='pt', padding=True, truncation=True)
with torch.no_grad():
    user_output = model(**tokenized_user)

# Mean pooling for user input embeddings
user_embedding = user_output.last_hidden_state.mean(dim=1).numpy()

# Compute cosine similarity between user input and faculty descriptions
similarity_scores = cosine_similarity(user_embedding, faculty_embeddings).flatten()

# Rank faculty based on similarity scores
faculty_df['Similarity'] = similarity_scores
sorted_faculty = faculty_df.sort_values(by='Similarity', ascending=False)

# Display the top faculty recommendations
top_faculty = sorted_faculty[['name', 'Similarity']].head(5)
print(top_faculty)
```

```
                    name  Similarity
21  Dr. Shashank Tamaskar    0.704869
17           Dr. Siddharth    0.695747
11      Dr. Sunita Chauhan    0.695186
9       Dr. Swagata Halder    0.679798
30         Dr. Rucha Joshi    0.676053
```

TRUTH TABLE - MEASURING ACCURACY

We created a **ground truth table** with manually identified and verified recommendations for various inputs.

Our model was tested on the same inputs and classified correct if 3/5 recommendations matched.

MODEL ACCURACY

```
# Calculate accuracy
accuracy = accuracy_score(true_labels_binary, correct_predictions)
print(f"Accuracy: {accuracy:.2%}")
```

```
Accuracy: 83.00%
```

USER FEEDBACK - TO CALCULATE PERFORMANCE

We also manually collected feedback from our peers to test the results of our model

Average score of **3.62/5 (72%)**

Low scores from:

- Prompts that were unrelated (Ex.Gaming)
- Uncovered feilds from data (Ex.Traffic management)

Name	Prompt	Feedback
Sauhard Sharma	Robotics and its applications	4
Ayush Seth	Renewable energy sources and their sustainability	5
Sarthak Sachdev	Virtual reality in enhancing classroom learning experiences	3
Rishi Vijaywargiya	Traffic management and road measures	2
Divith Narendra	Impact of AI on employment and job markets	4
Priyansh Desai	Online learning platforms and traditional education	3
Sanidhya Singh	Smart home technology and user privacy	3
Priyanshu Singhal	Dark web and cybersecurity risks	4
Aryaman Khandelwal	Financial technology in banking services	5
Shaurya Mann	eSports and gaming	1
Anshul Rana	Future of electric vehicles and transportation	2
Nikita Thomas	Neural networks and LLM	3
Rushiraj Gadhavi	Ethical hacking and cybersecurity defense	4
Suryansh Poonia	Computer Science and AI	4
Siddhant Deshpande	Finance with data science	5
Shubham Jain	Game theory	5
Avishi	Bioengineering and neuro science applications	4
Abhinav	Implementation of data structures	2
Malhar	Data mining and real life applications	4
Aanya	Applications of economic models	4
Anushka	Design and design thinking	5
	Average	3.6190476

FUTURE FORE SIGHT

1.Data Privacy and Ethics

Challenge: Expanding our dataset by scraping LinkedIn poses a challenge as the platform does not permit scraping activities.

Solution: We will employ web drivers to access LinkedIn data. We will ensure data privacy and compliance with LinkedIn's policies by anonymising user names and focusing solely on extracting area of expertise and interests, to enhance the training of our model.

2. Weighted Recommendations

Challenge: Currently, we don’t take into account the timeline of the work done by the professors. We also aren’t giving a priority queue to the user inputs.

Solution: Assign weights which are factored into our recommendations. The recent work done/ research interests will have higher weights. We aim to use data from either LinkedIn/their personal websites which are up to date and chronological.

THANK YOU!