1. **Automated Resume Evaluation System using NLP**

**Link :** [**https://ieeexplore.ieee.org/abstract/document/9036842**](https://ieeexplore.ieee.org/abstract/document/9036842)

**(Images are not clear!!!)**

1. Conversion Phase

**Example: Email Filtering System**

**Scenario**: A company receives numerous emails daily and needs an automated system to help filter spam and categorize important emails, such as those from clients or related to project updates.

**Steps**:

1. **Database**:
   * The system accesses a database containing incoming and historical emails.
2. **Structured vs. Unstructured Names**:
   * Emails are either structured (with clear fields like subject, sender, and body) or unstructured (like free-form text).
3. **Splitting into Tokens**:
   * **Action**: The system breaks down the email content into manageable pieces or tokens, typically words or phrases.
   * **Example**: "Urgent: Please review the attached report by EOD."
4. **Pre-tagging**:
   * **Action**: Basic attributes of tokens are identified; for instance, recognizing names, locations, time expressions, or action words.
   * **Example**: "Urgent" is identified as a priority indicator; "EOD" (end of day) as a time expression.
5. **Tagging Every Token**:
   * **Action**: Each token is tagged with more specific labels based on its content and context within the email.
   * **Example**: "Urgent" tagged as Priority: High, and "review" tagged as Action: Required.
6. **Split Based on Data Format**:
   * **Action**: The email is analyzed to identify which parts are plain text, which are hyperlinks, and which are possibly code or formatted text.
   * **Example**: A URL within the email is recognized and tagged as Link, while the rest remains as Text.
7. **Search for Every Token Match with Data Format**:
   * **Action**: Ensures each token fits the format expected; for example, checking if dates or URLs are valid.
   * **Example**: Validating that "EOD" matches a recognized time format for deadlines.
8. **Match**:
   * **Action**: If tokens match their expected formats and tags, the email is processed further; otherwise, it might be flagged for review.
   * **Example**: An email correctly tagged with high priority and required actions moves to the priority inbox.
9. **System Output**:
   * **Converted Data Results in Writing to the System**: Emails are sorted into categories like 'Priority', 'Spam', 'General Queries'.
   * **Converted into the Predifined (predefined) Schema**: Emails are stored in a structured format, making it easy to retrieve and analyze later.
10. **Extraction Phase**

**Example: Processing Customer Reviews**

**Scenario: An e-commerce platform receives a customer review: "The delivery was incredibly fast, and the product was exactly as described. Fantastic service!"**

* **Input String: The complete review text.**
* **Data Segmentation: The text is segmented into sentences:**
  + **"The delivery was incredibly fast."**
  + **"The product was exactly as described."**
  + **"Fantastic service!"**
* **Tokenization: Each sentence is broken down into tokens (words).**
  + **Sentence 1: ["The", "delivery", "was", "incredibly", "fast"]**
  + **Sentence 2: ["The", "product", "was", "exactly", "as", "described"]**
  + **Sentence 3: ["Fantastic", "service"]**
* **Feature Tagging: Words are tagged with parts of speech:**
  + **Sentence 1: ["The" (Determiner), "delivery" (Noun), "was" (Verb), "incredibly" (Adverb), "fast" (Adjective)]**
* **Pre-tagged Sentence: The entire sentence structure with tags.**
* **Entity Classification: No specific named entities here, but "delivery" could be tagged as a Service Feature.**
* **Recognized Entity: "Delivery" as Service Feature.**
* **Feature Extraction: Extract features such as speed ("fast") and quality of description ("exactly as described").**
* **Feature Recognition: Recognize positive sentiment from "Fantastic service!" and associate it with high customer satisfaction.**

1. **Filtering**

**The process of filtering resumes is predicated on comparing the job skills posted by the employer with the extracted data from the resumes. This process gives all the applicants who match the job description. To make the filtration process more efficient, a score is given to each resume to rank the applicant. Collaborative filtering is used to to predict the trend of selection [11].**

1. **Ranking**

** Data Ingesting:**

* **Action: The service collects data about movies, including genres, release dates, director, cast, viewer ratings, and reviews.**
* **Purpose: To create a comprehensive dataset that can be indexed and queried.**

** Indexer:**

* **Action: The data about each movie is organized into a searchable index that allows for quick retrieval based on different attributes (e.g., genre, popularity).**
* **Purpose: To facilitate efficient retrieval of movie information when a user makes a query.**

** Models:**

* **Action: Statistical models are developed to understand and predict user preferences and movie popularity.**
* **Purpose: To enhance the relevance of movie recommendations.**

** User Query:**

* **Action: A user enters a search term, such as "best sci-fi movies of the 2020s."**
* **Purpose: To find movies that match the user’s interests.**

** Topic Retrieval:**

* **Action: The system uses the indexed data to find movies that match the query.**
* **Purpose: To quickly gather a list of movies that are potentially relevant to the search term.**

** Ranking Model:**

* **Action: The retrieved movies are ranked according to a model that considers factors like relevance to the query, user ratings, and possibly the viewing history of the user.**
* **Purpose: To order the movies in a way that the most relevant results appear first.**

** Learning Algorithm:**

* **Action: The system analyzes how users interact with the recommended movies (e.g., if they choose to watch a recommended movie, how long they watch it, whether they like it, etc.).**
* **Purpose: To refine the recommendation algorithm based on real user feedback to improve future recommendations.**

** Training Data:**

* **Action: Historical data about user interactions with movie recommendations is used to train the machine learning models.**
* **Purpose: To continually improve the system by learning from past successes and failures.**

** Predictor:**

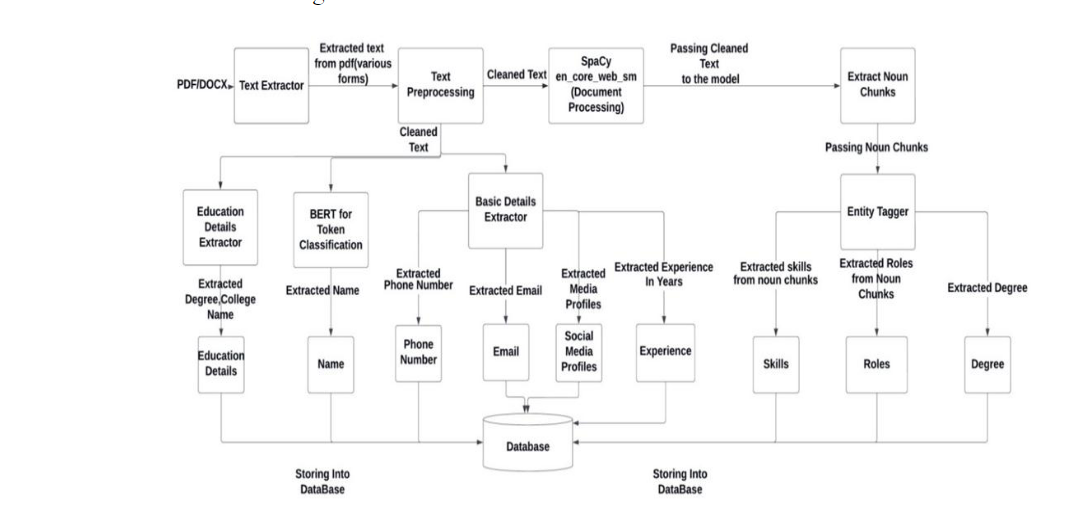
* **Action: Based on the refined models, the system predicts which movies the user will likely enjoy and displays them as recommendations.**
* **Purpose: To provide personalized movie recommendations that are likely to satisfy the user.**

1. **A smart resume screening tool for customized  
   shortlisting**

**Link :** [**https://www.researchgate.net/publication/373013406\_A\_smart\_resume\_screening\_tool\_for\_customized\_shortlisting**](https://www.researchgate.net/publication/373013406_A_smart_resume_screening_tool_for_customized_shortlisting)

* 1. **The model  
     works by taking the resume files in .doc, .docx, and .pdf formats.**
  2. **An Education Details Extractor and a Basic Details Extractor receive the cleaned text.  
     The Education Details Extractor returns the candidate's educational information. To extract  
     Indian names, the Bidirectional Encoder Representations from Transformers(BERT) model  
     is employed. The Basic Details Extractor gathers information such as phone numbers, email  
     addresses, and social network profiles.**
  3. **The Noun Chunks Extractor takes the cleaned text and extracts the nouns. The Entity  
     Tagger takes these nouns and classifies them as a skill, a role, or a degree. The nouns that  
     aren't relevant are ignored. The database stores all of the extracted information**

**Diagram:**

****

**3.1 Text Extraction and Preprocessing  
The resume text is extracted with the Python-based pdfminer program. It works with PDF  
and DOCX documents. The pdfminer automatically analyses the layout and extracts the  
text from the file. The pdfminer uses a lazy parsing mechanism, which means it only  
decodes information as needed**