ABSTRACT

Matching resumes with jobs is an important difficulty in recruitment systems, which consists of finding infor-mative information from resumes and job descriptions in order to determine suitability fit for the candidate. Traditional methods of address this challenge usually follow the identify-feature-extract-context phase in manual feature engineering, or people use existing systems that rely on keywords, which can be inefficient and may lack representation or context. This study proposes a unique high performing matching model for resume-job matching.

This Project proposes an intelligent systematic approach to match resumes to job descriptions and vice versa, which is advanced, fast, accurate, and efficient. The system will employ a scoring model to evaluate resumes measure based on many factors including: skills, experience, education, and location. The system will use some custom-trained Word2Vec based on the CBOW (Continuous Bag of Words) method to provide the word embeddings for the textual data, and then the system will then use cosine similarity to calculate the similarity score between the resumes and job description. The Gale–Shapley stable marriage algorithm (the thermometer) will be used to find optimal pairs between the candidates and the job roles.

The smart resume-job matching system will handle large volumes of data efficiently; rank candidates and provide a stable matching opportunity for the candidate and job opportunities, which would be beneficial for both the employer and job seeker. The use of a custom trained Word2Vec model gave the data contextual understanding compared to using existing trained embeddings, which improved the accuracy of the results.

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

Natural Language Processing (NLP) aims to enable machines to read, understand, and generate human language in a useful way. NLP has been one of the busiest domains for deep learning research and development, with applications ranging from sentiment analysis to document classification, automated conversation (chatbots), and generating language. Whatever the immediate task, the underlying tasks of text classification and generation are fundamental in allowing machines to label the input using contextually appropriate classes/categories and generate decent language outputs.

Recruitment systems are trying to help fill the gap between job opportunities and candidates by reading and interpreting textual documents such as resumes and job descriptions. With the rise of Natural Language Processing (NLP), automated resume parsing and profile matching has become a common tool in modern hiring systems. NLP methods extract structured information, determine relevance, and produce recommendations that act like a human, using the area of knowledge as an objective function.

However, traditional keyword matching systems typically do not display a capability of identifying the semantic and contextual relationship between job particularities and candidate qualifications. Organizations receive thousands of applications. As such, the need for scalable, efficient, and context aware matching models are required. A well trained matching system must account for nuances of skill, experience, education, and job roles and be able to align these characteristics with the expectations in job descriptions of these characteristics.

This research provides an intelligent semantic framework with an algorithm combining semantic similarity scoring using Word2Vec embeddings, and the stable matching, Gale-Shapley algorithm in order to determine the best / fair pairing of resumes with job descriptions. Word embeddings provide the deep contextual associations with the terms for the semantic matching, while the pairing algorithm provides stable and mutually satisfactory candidate-job matches. The two components provide a scalable and accurate method of automated talent acquisition.

1.2 PROBLEM STATEMENT

This Project is an intelligent system automating the hiring process through resume to job description matching and job description to resume matching. The proposed system has a scoring model for resumes using various aspects, including skills, experience, education, and location of the candidates. A custom Word2Vec model (CBOW - Continuous Bag of Words) is used to provide word embeddings to the textual data. Cosine similarity is then calculated on resumes with job descriptions, returning a single profile-scored similarity. The Gale-Shapley stable matching algorithm is used to ensure an ideal matching of candidate-to-role.

The system is designed to accommodate and process large data sets efficiently, produce a ranking of candidates for job descriptions, and provide a stable matching that benefits both employers and job seekers. Experimental results identified the effectiveness of scoring model presented, whereby larger similarity scores suggest better alignment to job requirements. The implementation of custom Word2Vec model allowed the system to better understand context than using standard pre-trained word embeddings, from which improved results were derived.

Future work will investigate the usage of advanced embedding techniques from models like BERT or ELMO, and the increase in data in the corpus to continue to improve the model. The deployment of this model demonstrates the opportunities to create better procedures for recruitment through the automation of resume parsing, scoring, and matching that are accurate and reliable. .

1.3 OBJECTIVES

The main objectives of this work are as follows:

- To develop an automated system that semantically analyses and matches resumes with job descriptions using NLP techniques.
- To implement a scoring mechanism based on **Word2Vec embeddings** that evaluates similarity across key attributes like skills, experience, designation, and location.
- To design and apply a **Gale-Shapley stable matching algorithm** to ensure optimal and conflict-free pairings between candidates and job openings.
- To preprocess and structure unformatted resume and job description data for accurate entity extraction and scoring.
- To evaluate the effectiveness of the scoring and matching system using real-world resume datasets and job profiles.
- To build a scalable and intelligent framework that reduces manual screening effort and improves the quality of recruitment outcomes.

CHAPTER 3
METHODOLOGY

3.1 INTRODUCTION

Resume and job matching is becoming an increasingly important part of modern recruiting systems, where employers are required to assess thousands of resumes and identify which candidates are potentially best suited to a specific set of job qualifications. In addition, applicants' resumes or qualifications are assessed using keyword-based approaches that sometimes disregard the semantic relationships between jobs and candidates. This project described creates an intelligent, NLP-based system that automates the matching process of resumes to jobs by evaluating the textual data for the semantic relationship and generating structured scores to assess compatibility.

The system uses Word2Vec embeddings trained using a Continuous Bag of Words (CBOW) neural model to create embedded representations of resumes and job specifications in continuous dense vector space that provide semantic aware contextual meaning for the words expressed in each resume and job description. To assess the closeness of a candidate's profile to the job description, the embeddings' cosine similarity between resumes, and job specifications yield an assessment of closeness for criteria such as skills, experience, designation and location. In assessing close match pairs of jobseekers and job roles at the worker level, the Gale-Shapely algorithm used in a stable matching context allows for a fair, and optimal jobseeker-job role assignments, such that there are no better pairs outside the match pairs provided. The iteration through the matching algorithm forms the basis of the jobseeker-job role recommendations, which are reliable, and likely free of personal bias.

3.2 Gale Shapely Algorithm

The Gale-Shapley algorithm, otherwise known as the Deferred Acceptance Algorithm, was characterized by David Gale and Lloyd Shapley in 1962 and used to solve the Stable Marriage problem. The fundamental idea is to find a stable matching between two equally-sized groups of participants (usually referred to as men and women) each of whom ranks all members of the opposite-group by preference. A matching is stable if there are no two people who would rather be matched with each other than their assigned partners. This guarantees that no couple will want to deviate away from the assigned match, allowing them to remain content in the system long-term. The algorithm works by having two sides (the men and women) make proposals and tentative acceptances in an iterative fashion. The men will start by proposing their preferred choice. Each woman will then hold her best offer that conforms to her preferences and reject the rest. In the second round, the rejected men will propose to their preferred choice. This continues until everyone is matched, and no one can further propose. At this point, the matching is complete and stable.

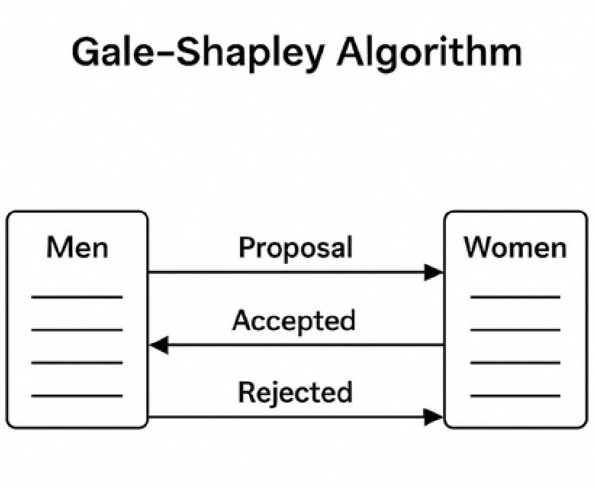


Fig 3.1 Architecture of Gale Shapley Algorithm Model

One of the main properties of the Gale-Shapley algorithm that stands out is that it always yields computing a stable matching independent of the initial preference input. Additionally, it is optimal for the proposing side that means that each proposer receives the best partner they could receive in any stable matching. Conversely, it is pessimal for the receiving side which ends up with their worst acceptable partner of all the stable outcomes. Therefore, this does a good job highlighting the trade-off of which side should have the proposing role in applications.

Key Tasks of Gale Shapley Algorithm are :

1. **Query-Document Matching** : Match search queries with the most relevant documents based on mutual relevance rankings.
2. **Intent Detection and Utterance Matching** : Pair user utterances with the most appropriate intent classes, respecting both user and system preferences.
3. **Question-Answer Pairing** : Match user questions with high-quality answers when both questions and answers have ranked preferences (e.g., in QA systems).
4. **Sentence Alignment in Summarization** : Align summary sentences with the most informative source sentences in extractive or abstractive summarization tasks.
5. **Phrase Alignment in Machine Translation** : Match phrases from source and target languages based on translation likelihoods and contextual preferences.

The algorithm is also **efficient** , with a worst-case time complexity of $O(n^2)$, where n is the number of participants in each group. This makes it practical for use in large-scale real-world systems.

3.3 WORKING MECHANISM

3.3.1 Workflow

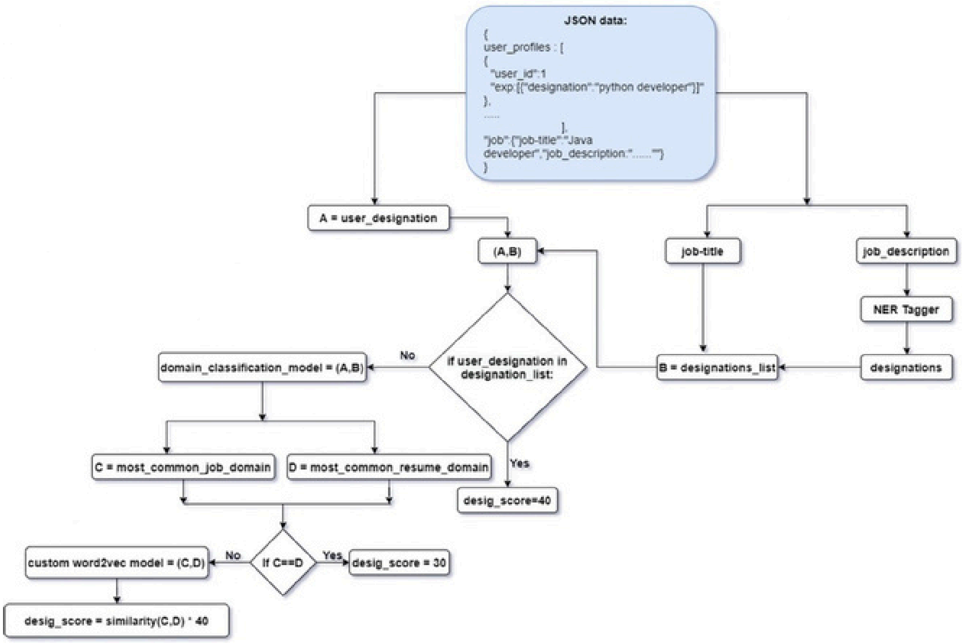

JSON data:
{
user_profiles : [
{
"user_id":1
"exp:[{"designation":"python developer"}]"
},
......
],
"job":{"job-title":"Java developer","job_description:"......""}
}
A = user_designation
(A,B)
job-title
job_description
NER Tagger
designations
B = designations_list
if user_designation in designation_list:
No
Yes
domain_classification_model = (A,B)
C = most_common_job_domain
D = most_common_resume_domain
desig_score=40
If C==D
No
Yes
custom word2vec model = (C,D)
desig_score = 30
desig_score = similarity(C,D) * 40

Fig 3.2: Workflow of the Model

3.3.2 Dataset

The NYC dataset, published on Data.gov, offers a full list of jobs anywhere under New York City's government. This dataset is released to the public with a wealth of different information like job titles, agency names, salary, closing dates for applications and more. The dataset is very useful to job seekers, academics and policy makers. The dataset is updated on a basis and has multiple formats, including CSV, JSON, and XML. Providing users with more choices for their analytical tool of choice.

Dataset Overview

- **Title:** NYC Jobs
- **Publisher:** City of New York
- **Last Updated:** Varies (check the page for the latest version)

- **Format:** CSV, JSON, XML, RDF
- **License:** Public Domain (U.S. Government)

Features of the datasets:

The dataset includes structured data on job openings, including:

1. Job Details:
   - Job ID – Unique identifier for the posting.
   - Agency – NYC department offering the job (e.g., "DEPARTMENT OF HEALTH").
   - Business Title – Job title (e.g., "Administrative Manager").
   - Civil Service Title – Official civil service classification.
2. Location & Salary:
   - Work Location – Borough or address (e.g., "Manhattan").
   - Salary Range – Min/max salary (e.g., "$50,000 - $60,000").
3. Requirements & Logistics:
   - Job Description – Responsibilities and duties.
   - Minimum Qual Requirements – Education/experience needed.
   - Preferred Skills – Desired qualifications.
   - Full-Time/Part-Time – Employment type.
4. Application Process:
   - Posting Date – When the job was listed.
   - Post Until – Deadline to apply.

Potential Use Cases

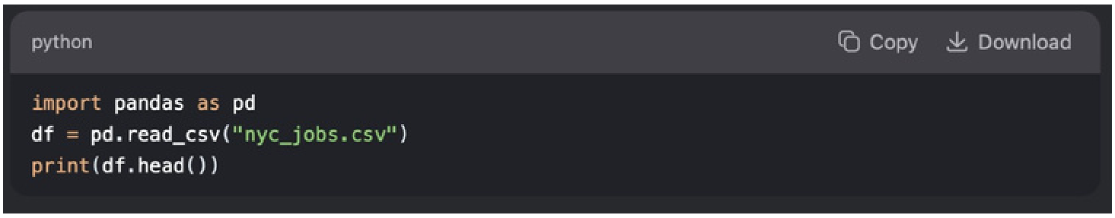
1. **Job Market Analysis**
   - Determine the most searched jobs or agencies with the most steady hiring.
   - Compare salary ranges across sectors (e.g., healthcare vs. sanitation).
2. **Public Policy Research**
   - Examine hiring equity (e.g., geographic distribution of jobs).

- Track changes in civil service requirements over time.

3. **ML/NLP Applications**
   - Use Business Title or Job Description to classify jobs.
   - Predict salary ranges based on skills/requirements.
4. **Job Seekers' Tool**
   - Build a dashboard filtering jobs by salary, location, or agent

How to Access

1. Download Options:
   - Direct CSV/JSON links are available on the page.
   - API access via Socrata (if supported).
2. Preview:
   - Use tools like Pandas (Python) or Excel to explore:



```python
import pandas as pd
df = pd.read_csv("nyc_jobs.csv")
print(df.head())
```

Fig 3.3 Python Code to View DataSet

3.3.3 Text Preprocessing:

Text preprocessing is the first and important stage in the pipeline, where we will make sure that all the textual data from resumes and job descriptions is clean and structured, semantically meaningful, before being converted into word embeddings using the Word2Vec model. The complete pre-processing routine follows:

1. **Text Cleaning**

Unstructured text from resumes and job postings often contains irrelevant symbols, formatting issues, and noise (like punctuation, numbers, or stopwords). This step involves:

o Removing special characters and numbers o Lowercasing the text

o Removing stopwords (e.g., “the”, “is”, “and”)

o Removing or handling irrelevant metadata

2. Tokenization

The cleaned text is then broken into individual tokens (words or phrases). This step helps in analyzing and embedding each word independently.

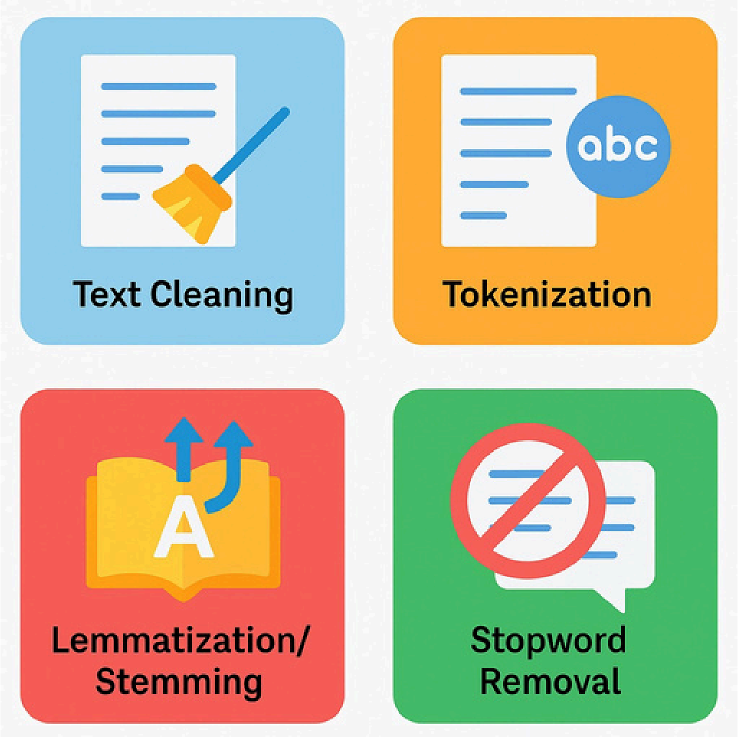


Fig 3.4: Text Preprocessing

1. Lemmatization/Stemming *(implied)*

Words may be reduced to their root forms to normalize variants (e.g., “running”, “ran”, “runs” → “run”), ensuring consistency across similar terms.

2. Sentence Rewriting and Normalization

Sentences or bullet points in resumes and job descriptions are rewritten or normalized to eliminate duplication and maintain consistency in language. This helps the Word2Vec model to better identify semantically similar content.

3. **Entity Recognition and Filtering** *(referenced indirectly in scoring and matching)*
   Named Entity Recognition (NER) may be applied to identify and tag key items such as skills, job titles, locations, or qualifications, which are later used in scoring.

4. **Vectorization Preparation**

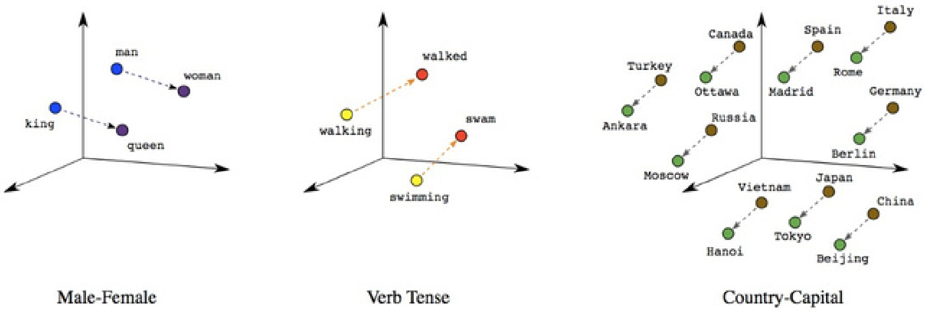
   The processed text is now in a structured format that can be embedded using the **CBOW Word2Vec** model. Average embeddings are calculated to represent full documents (resumes or job descriptions) as single vectors.

Overall, text preprocessing transforms raw, unstructured job and resume data into clean, consistent input ready for downstream tasks like **semantic scoring** and **Gale-Shapley-based matching**. It plays a vital role in the system's ability to understand the **true meaning and context** behind words and phrases in both resumes and job listings.

3.3.4 Word Embedding:

Word embedding is a way of representing words as dense vectors of real numbers as part of **Natural Language Processing (NLP).** These word embedding vectors are able to represent **semantic meaning** and context, meaning that we can tell the machine what words are "similar" to each other in a way that goes beyond simply matching words.

Compared to other representations, such as a sparse representation of words with **one-hot encoding** (where it only indicates whether or not a particular word is in the corpus), word embeddings represent similarities, so that the words "engineer", "developer", and "programmer" would be clustered closer together in vector space.



Male-Female

Verb Tense

Country-Capital

Fig 3.5: Word Embedding

Word embedding is used to convert the textual content of resumes and job descriptions both into numerical formats that can be compared and scored.

The process is implemented using the **Word2Vec algorithm**, specifically the **CBOW (Continuous Bag of Words)** model.
Model Used: Word2Vec (CBOW)

- The project uses the **Word2Vec model**—specifically the **Continuous Bag of Words (CBOW)** architecture.
- In **CBOW**, the model predicts a target word based on its surrounding context words.
- This approach is efficient and well-suited for capturing semantic similarities in textual data like skills, experience, and job roles.

WHY CBOW (CONTINUOUS BAG OF WORDS)?

CBOW is one of the two architectures used in Word2Vec (the other being Skip-Gram).

Here’s why **CBOW** was chosen:

- **Speed and Efficiency**: CBOW is faster and suitable for larger datasets like the 12,000+ resumes used in this project.

- **Contextual Prediction**: CBOW learns to predict a word given its context, which helps capture word usage and meaning in resume text.
- **Smooth Representations**: It results in **generalized embeddings** that work well when averaging over entire documents (like a resume).

STEPS FOLLOWED IN WORD EMBEDDING

1. Corpus Creation

A large dataset of 12,000 resumes is used to train the model. This ensures that embeddings are domain-specific, capturing job-related terminology more accurately.

2. Training the Word2Vec Model

- The **CBOW architecture** is used to train the Word2Vec model on this corpus.
- Training is done using the **Gensim** library in Python, which is efficient and supports model tuning.

3. Generating Embedding Vectors

- Each word in the resume or job description is converted into a dense vector using the trained model.
- These vectors typically have **100 to 300 dimensions**, depending on the model configuration.

4. Document Vector Representation

- For a full resume or job description, the system computes the **average** of all word vectors in that document.
- This results in a **single vector per document**, which summarizes the semantic meaning of the entire resume or job description.

STEPS FOLLOWED IN WORD EMBEDDING

1. Corpus Creation

A large dataset of **12,000 resumes** is used to train the model. This ensures that embeddings are **domain-specific**, capturing job-related terminology more accurately.

2. Training the Word2Vec Model

- The **CBOW architecture** is used to train the Word2Vec model on this corpus.
- Training is done using the **Gensim** library in Python, which is efficient and supports model tuning.

3. Generating Embedding Vectors

- Each word in the resume or job description is converted into a dense vector using the trained model.
- These vectors typically have **100 to 300 dimensions**, depending on the model configuration.

4. Document Vector Representation

- For a full resume or job description, the system computes the **average** of all word vectors in that document.
- This results in a **single vector per document**, which summarizes the semantic meaning of the entire resume or job description.

3.3.5 SIMILARITY MEASUREMENT

Once both resumes and job descriptions are embedded as vectors, the **cosine similarity** is computed between them.

- **Cosine Similarity = 1** → Perfect match
- **Cosine Similarity = 0** → No similarity
- The result is a **numerical score** used in the next phase for matching.

This similarity score becomes an input for the **Gale-Shapley algorithm**, which uses it to create stable, optimal job-candidate pairings.

BENEFITS OF USING WORD EMBEDDING :

1. **Semantic Understanding**

   Captures context, meaning, and relatedness of terms in resumes and jobs, unlike keyword-based matching.

2. **Domain Adaptability**

   Trained on resume data, so it learns relationships specific to hiring (e.g., “developer” ~ “programmer”).

3. **Efficient Comparison**

   Converts long and complex documents into compact vector forms, allowing fast and scalable comparison.

4. **Improved Matching Accuracy**

   Helps identify candidates who may be relevant even if exact keywords don’t match.

FUTURE SCOPE MENTIONED IN THE REPORT

While Word2Vec offers strong performance, the report also proposes **replacing Word2Vec with more powerful models** like:

- **BERT** (Bidirectional Encoder Representations from Transformers)
- **ELMo** (Embeddings from Language Models)

These models also offer context-sensitive embeddings at the sentence level considering polysemy and increasing overall performance.

The Model, in this case, utilizes Word2Vec and CBOW architecture to create word embeddings from resume data and the job descriptions data. These word embeddings are subsequently averaged together into document level vectors, which are then compared using cosine similarity. This method provides a semantic basis for scoring and matching and forms a foundational pillar of the resume to job matches system allowing for fair, accurate, and scalable recruitment automation.

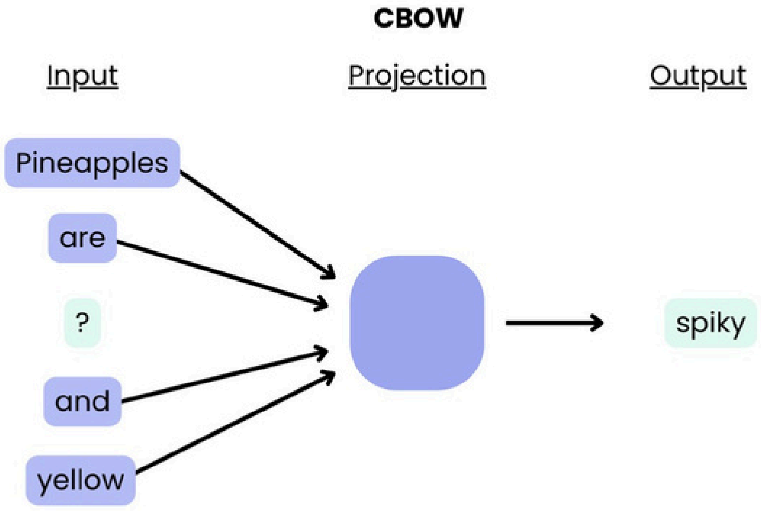


Fig 3.6: Common Bag of Word

3.3.6 Scoring Mechanism :

The scoring mechanism is a key component in determining how well a resume (CV) fits a job description (JD). The aim of the scoring mechanism is to assign a numerical compatibility score to each resume- job pair to enable stable matching using the Gale-Shapley algorithm.

OVERVIEW OF THE SCORING SYSTEM

The scoring mechanism is designed to:

- **Quantify the similarity** between resumes and job descriptions.
- Incorporate multiple evaluation criteria to ensure a comprehensive assessment.

- Feed these scores into the **Gale-Shapley algorithm** for optimal and stable matching.

TYPES OF SCORING SCENARIOS

The system supports multiple scoring configurations:

1. **One Job Description → Multiple CVs**
   - o Score how well each resume fits a single job role.
2. **One CV → Multiple Job Descriptions**
   - o Score how well a candidate fits various available jobs.
3. **One CV ↔ One Job Description (Content-wise)**
   - o Direct comparison between a single resume and job post using semantic similarity.

TECHNIQUES USED FOR SCORING

1. Word Embedding with Word2Vec

- The textual data is converted into vectors using a **Word2Vec model (CBOW)** trained on a corpus of resumes.
- Both resumes and job descriptions are embedded into a **shared vector space**.
- Each document is represented by the **average of word embeddings** for its content.

2. Cosine Similarity

- The similarity between two vectors (resume and job description) is calculated using **cosine similarity**, a standard method to assess the angle between vectors in high-dimensional space.
- A higher cosine similarity indicates a better match.

COMPONENTS OF THE TOTAL SCORE

Criterion	*Weight*
Designation Score	40%
Skills Score	25%
Experience Score	15%
Progress Score	15%
Distance Score	5%
Total	100%

The final score is a **weighted sum of individual scores** from several criteria. Each component represents a different dimension of compatibility:

Table 3 : Percentage Distribution Criteria

Explanation of Each Component:

1. **Designation Score (40%)**

    o Compares the job title in the CV with the title in the job description.

    o Uses NER (Named Entity Recognition) to detect titles like *"Software Engineer"*, *"Manager"*, etc.

    o Hierarchical mapping is used (e.g., Junior < Mid < Senior) to assess the match.

2. **Skills Score (25%)**

    o Extracted from both CV and JD using keyword/entity recognition.

    o Matched using embedding-based similarity to assess skill relevance and coverage.

3. **Experience Score (15%)**

    o Considers years and type of experience (e.g., relevant domain experience).

    o May involve rule-based checks (like duration from previous job entries).

4. **Progress Score (15%)**

    o Attempts to quantify career progression (e.g., intern → developer → senior developer).

    o Uses rule-based methods to detect upward mobility or consistency in roles.

5. **Distance Score (5%)**

    o Considers geographical distance between candidate location and job location.

    o Lower distance = higher score, assuming proximity preference.

OUTPUT FORMAT

- The results are returned as a **JSON object** containing:

    o A list of **user profiles (CVs)** with their scores against job descriptions.

    o A list of **job descriptions**, each with a ranked list of suitable candidates based on the scores.

ROLE IN MATCHING

- These scores act as **input preferences** for the **Gale-Shapley algorithm**.

- Each side (CV and job description) ranks the other based on the scores.
- The algorithm uses these rankings to generate a **stable match set**, where:
    - No resume and job description would prefer to be matched with each other over their assigned pair.

ADVANTAGES OF THIS SCORING SYSTEM

- **Comprehensive**: Takes multiple criteria into account, not just keywords.
- **Semantic**: Uses embeddings for deep textual understanding.
- **Flexible**: Can handle both one-to-many and many-to-one scenarios.
- **Fair**: Ensures optimal pairing with stable matching logic.

3.3.7 Gale Shapley Algorithm:

In 1962, the mathematicians David Gale and Lloyd Shapley introduced the Gale-Shapley algorithm, also known as the Stable Marriage Algorithm, to solve the stable matching problem, or the process of pairing the elements of two sets (“men” and “women” in the traditional sense;) based on their rankings/preferences such that no two individuals of separate groups would both rather be matched with each other than with their partner. The algorithm guarantees a stable matching of the original two sets (while one is the proposers and the other is the acceptors). Initially, one group will propose to the members of the other group in the order of their preferences. The acceptors will only allow (tentatively) or will reject the proposals based on their preferences (including allowing or proposing to multiple individuals). The process will continue until every proposer is matched, meaning that no pair would prefer each other over their partner.

HOW GALE-SHAPLEY WORKS (BASIC STEPS)

1. **Initialize**: All members of both groups are unmatched.
2. **Propose**: An unmatched proposer proposes to the highest-ranked acceptor on their list.
3. **Decision**:
   - o If the acceptor is unmatched, they accept the proposal.
   - o If already matched, they compare the current match with the new proposer:
     - ▪ If they prefer the new proposer, they accept and reject the current one.
     - ▪ If not, they reject the new proposal.
4. **Repeat**: The rejected proposer moves on to the next preference.
5. **Terminate**: When all proposers are matched, the algorithm ends with a **stable set of matches**.

HOW GALE-SHAPLEY IS USED IN THIS MODEL

The Gale-Shapley algorithm is **repurposed** to match **resumes (CVs)** with **job descriptions (JDs)** in a fair and optimal way.

Roles Reassigned

- **Group A (Proposers)**: Resumes (CVs)

  Candidates apply to jobs based on compatibility scores (their “preference list”).

- **Group B (Acceptors)**: Job Descriptions (JDs)

  Jobs rank candidates based on how well they match the role (job’s “preference list”).

These preferences are **derived from the scoring mechanism**, which uses **Word2Vec embeddings** and **cosine similarity** to calculate how well a CV matches a JD.

Process Adaptation for Resume Matching

1. **Preference Lists Creation**:
   - o For each **CV**, rank all **JDs** by similarity score (from highest to lowest).
   - o For each **JD**, rank all **CVs** by similarity score (from highest to lowest).
2. **Proposal Process**:
   - o Each resume (CV) "proposes" to the top-ranked job.
   - o The job either:
     - ▪ Accepts the CV if it is unmatched.
     - ▪ Compares the new CV to the current match and **accepts the better one**.
     - ▪ Rejects the lesser one (who then proposes to the next job).
3. **Stable Match Output**:
   - o Matching continues until **every resume is matched** or **all preferences are exhausted**.
   - o The final result is a **set of stable job-candidate pairs**.

Input and Output Structure

- **Input**:
  - o JSON files containing:
    - ▪ CVs with their scores for each job.
    - ▪ JDs with their scores for each resume.
  - o These scores act as preference indicators.

- **Output**:
  - A single, stable matching set that **maximizes compatibility** and **ensures stability** (no unmatched CV–JD pair prefers each other over their assigned match).

BENEFITS OF USING GALE-SHAPLEY IN THIS SYSTEM

Feature	Benefit
Stability	Prevents “unfair” matchings where better pairings are left unmatched.
Optimal Matching	Every CV is matched to its best possible JD (based on preferences).
Scalability	Efficiently handles thousands of resumes and jobs.
Fairness and Transparency	Both sides (candidates and recruiters) have a say in match outcomes

Table 2 : Benefits of Gale Shapley Algorithm

In this project, the Gale-Shapley algorithm is used to realize stable and optimal matches between resumes and job descriptions based on semantic similarity scores. The resumes and job descriptions are treated as two sets with mutual preferences, calculated from vector representation into a score, and a mechanism that operates in a systematic iterative fashion, matching them in a fair way. This is so that the hiring system not only assists with automation of the matching component, but in a way that is mathematically stable, unbiased, and efficient.

3.3.8 Conclusion :

It was successfully demonstrated that a scalable and intelligent system for automated resume/job description matching based on Natural Language Processing (NLP) and algorithm optimization techniques exists. Word2Vec-based word embeddings were used to create cosine similarity scores for resumes/job descriptions to semantically assess congruity between resumes and jobs descriptions regardless of exact keyword matches.

Then, based on those similarity scores, the Gale-Shapley algorithm was used to create stable and fair matchings between job seekers and job openings. The scoring metrics were also weighted to enhance the rankings of job seeker profiles, or job postings based on skill, experience, designation, location, and progression, so that the model closely mirrored the contexts in which real hiring decisions are made. The overriding conclusion is that a custom-trained Word2Vec model performed very well on the dataset collected, and embeddings with contexts leads to the most relevant matches. Additionally, the system's modular design also means it is possible to modularly upgrade it in the future, for example, to more transformative models including BERT or ELMo, and continue to expand the dataset size to better improve generalization and accuracy.

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Scoring Results :

The **Scoring Output** is the core computational result of the system, representing how well a candidate's resume aligns with a given job description. Each resume–job pair is scored based on multiple **independent attributes**, which are extracted and evaluated using **NLP techniques**, **semantic similarity**, and **rule-based logic**.

These scores are **weighted and aggregated** to compute a **final compatibility score**, which is later used in the **Gale-Shapley stable matching algorithm** to generate optimized job–candidate pairings.

Purpose of Scoring Output

- To **quantify the match** between a CV and a job role using multiple relevant dimensions.
- To ensure **fair and comprehensive evaluation** of both technical and contextual information.
- To provide a **ranking basis** for job recommendation and candidate filtering

Component	Weight (%)	Description
Designation Score	40	*Measures title alignment between resume and job description*
Skills Score	25	*Compares relevant skills extracted via NLP and embeddings*
Experience Score	15	*Evaluates work experience and industry alignment*
Progress Score	15	*Analyzes career trajectory and promotion patterns*
Distance Score	5	*Considers geographic proximity between candidate and job location*

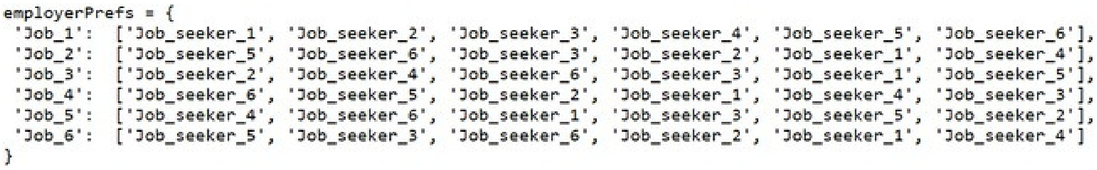
Table 3 : Weight Distribution.

These two datasets arrive in our system in the form of json files, so we will use the keyword to denote their locations. The first keyword contains the job description information and then all of the scores for each requirement. This score indicates the final score for a particular job description, which has defined requirements.

Since the scoring mechanism in the dataset is critical to the whole matching system, it provides a metric on the relationship between both resume and job descriptions. Specifically, it analyzes resumes against some key things such are skills, designation, experience, and location, producing a cumulative score for each resume-job pair. That composite score indicates the likelihood of successful outcome to the job description-resume matches, and will enter the Gale-Shapley algorithm.

How Scores Are Calculated

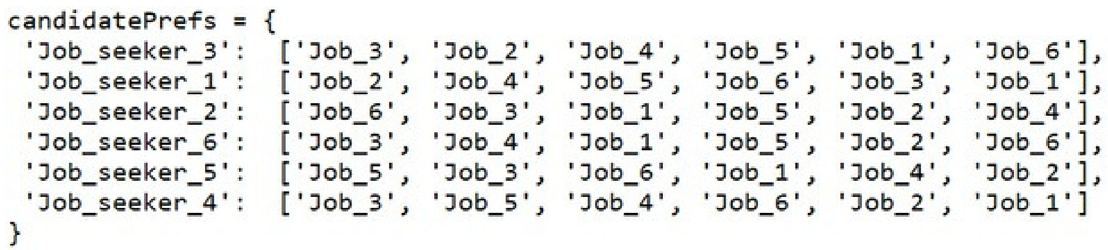
- Each feature is **extracted** from resumes and job descriptions after preprocessing (tokenization, cleaning, and entity recognition).
- **Word embeddings** (trained using CBOW Word2Vec) are used to **vectorize** skills, titles, and other textual attributes.
- **Cosine similarity** is applied to compare resume vectors with job vectors.
- Rule-based logic is used for fields like experience and location.
- Scores are **normalized to a 0–100 scale**, and then **aggregated using the defined weights**.



```
employerPrefs = {
 'Job_1':  ['Job_seeker_1', 'Job_seeker_2', 'Job_seeker_3', 'Job_seeker_4', 'Job_seeker_5', 'Job_seeker_6'],
 'Job_2':  ['Job_seeker_5', 'Job_seeker_6', 'Job_seeker_3', 'Job_seeker_2', 'Job_seeker_1', 'Job_seeker_4'],
 'Job_3':  ['Job_seeker_2', 'Job_seeker_4', 'Job_seeker_6', 'Job_seeker_3', 'Job_seeker_1', 'Job_seeker_5'],
 'Job_4':  ['Job_seeker_6', 'Job_seeker_5', 'Job_seeker_2', 'Job_seeker_1', 'Job_seeker_4', 'Job_seeker_3'],
 'Job_5':  ['Job_seeker_4', 'Job_seeker_6', 'Job_seeker_1', 'Job_seeker_3', 'Job_seeker_5', 'Job_seeker_2'],
 'Job_6':  ['Job_seeker_5', 'Job_seeker_3', 'Job_seeker_6', 'Job_seeker_2', 'Job_seeker_1', 'Job_seeker_4']
}
```

Fig 4.1: Json Representation of One JD with Multiple CV

Similarly, other keywords include data from each resume, with each individual score inall professional explanations. This score is provided manually by the user for each job description.



```
candidatePrefs = {
 'Job_seeker_3':  ['Job_3', 'Job_2', 'Job_4', 'Job_5', 'Job_1', 'Job_6'],
 'Job_seeker_1':  ['Job_2', 'Job_4', 'Job_5', 'Job_6', 'Job_3', 'Job_1'],
 'Job_seeker_2':  ['Job_6', 'Job_3', 'Job_1', 'Job_5', 'Job_2', 'Job_4'],
 'Job_seeker_6':  ['Job_3', 'Job_4', 'Job_1', 'Job_5', 'Job_2', 'Job_6'],
 'Job_seeker_5':  ['Job_5', 'Job_3', 'Job_6', 'Job_1', 'Job_4', 'Job_2'],
 'Job_seeker_4':  ['Job_3', 'Job_5', 'Job_4', 'Job_6', 'Job_2', 'Job_1']
}
```

Fig 4.2: Json Representation of One CV with Multiple JD

Values in both of these units are ordered within the descending order, i.e. the most desired CV or JD is at first. but, this facts represents a couple of strong set of preparations. And this is wherein we employ the set of rules that later gives a unmarried set of arrangement for each CV or activity-description

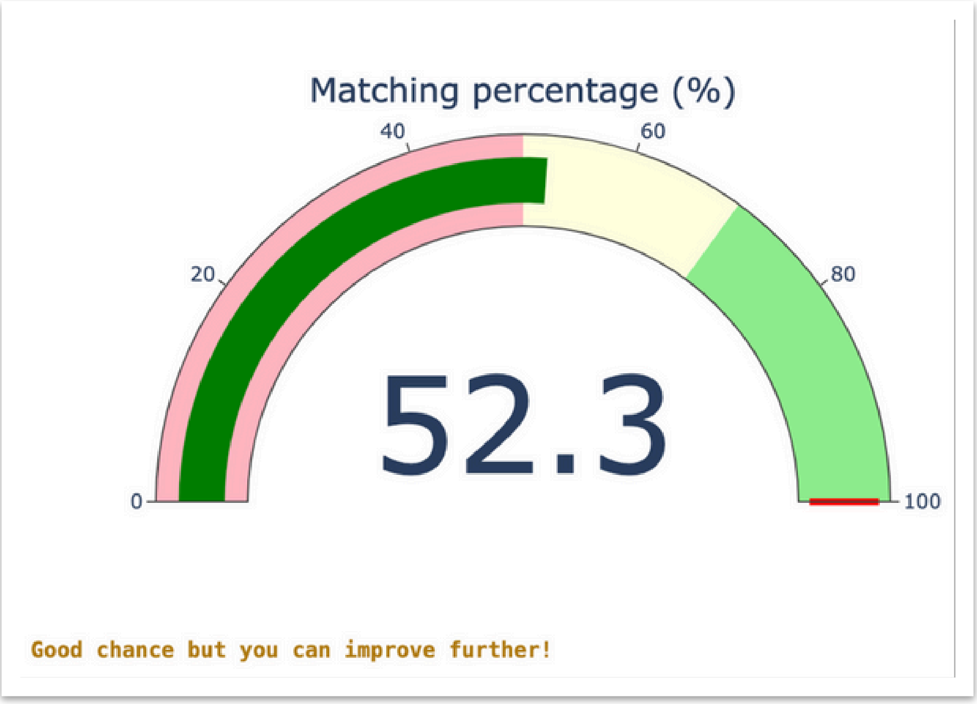


Fig 4.3: Matching Percentage of Resume to Job Description

4.2 CONCLUSION

The Resume and Job Matching System presented establishes an automated evaluation of candidates and jobs by leveraging semantic similarity and stable matching. In particular, the use of Word2Vec embeddings in conjunction with a weighted scoring mechanism can provide comprehensive sophisticated context understanding, and candidates can be fairly evaluated.

The system assesses for both matches considering several relevant features and also provides outputs based on a structured format (e.g., JSON) which can assist in integration. The Gale and Shapley stable matching algorithm enables optimal assignment of candidates to jobs without any conflicts. The evaluation values associated with attributes of skills, experience, and designation enable interpretable scores which gives the system a level of adaptability for practical use whilst maintaining a transparent process.

This system framework is transferable for usage in applicant tracking systems (ATS), career portals, or enterprise level human resource management tools, facilitates a holistic system that could operate on a large scale and potentially eliminate biases from modern hiring procedures. The resume and job matching system framework could be improved further with the combination of transformer based embeddings (e.g., BERT) and learning based classifiers to increase the accuracy of predicting the match and the levels of adaptation.

CHAPTER 5
CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

This proposed Resume and Job Matching System supports the automation of evaluating a candidate in relation to a job, by utilizing semantic similarity and stable matching algorithms. Using Word2Vec embeddings as a textual representation (with weighted scores) allows for deeper contextualizing within the matches and ensure there is fairness in the candidate rankings.

The system groups matches utilizing multiple relevant factors and returns structured JSON outputs to aid in integration as well. In addition the Gale-Shapley Algorithm guarantees that the candidates were matched to job roles in the best possible way with no conflicts. A number of the evaluation categories (for example: skills, experience and designation) provide reasonably interpretable scores to rank candidates in the rankings that made an intelligent and interpretable use of the system.

An expandable solution such as this could contribute strongly to ATS Platforms, career portals or enterprise HR tools to offer a unbiased, scalable solution to modern day hiring. Future enhancements could include incorporating transformer-based (BERT) embeddings, and finding and utilizing learning classifiers to improve match prediction accuracy and flexibility.

5.2 FUTURE SCOPE

- **Contextual Embeddings**: Integrating advanced models like BERT, RoBERTa, or ELMo can improve the semantic understanding of resumes and job descriptions by capturing deeper contextual relationships between words and phrases.

- Supervised Learning for Match Prediction: Training machine learning classifiers on historical match data can allow the system to learn optimal match patterns and improve predictive performance beyond rule-based scoring.
- Feedback-Based Learning: Implementing feedback loops where recruiter inputs (accepted/rejected matches) are used to refine future recommendations through reinforcement learning or continuous retraining.
- Robust Resume Parsing: Incorporating intelligent document parsing frameworks (like spaCy, LayoutLM, or Tika) can enhance the system's ability to handle unstructured, noisy, or incomplete resume formats.
- Domain-Specific Customization: Tailoring models for specific industries (e.g., healthcare, IT, finance) using domain-specific vocabularies and weighting schemas can boost accuracy for specialized roles.
- User Interface and ATS Integration: Developing intuitive dashboards and API-based interfaces will allow integration into applicant tracking systems (ATS) for real-time, scalable deployment.
- Scalable Cloud Deployment: Hosting the system on scalable cloud platforms with distributed processing capabilities will support batch processing of large volumes of resumes and job descriptions.