BIRMINGHAM CITY UNIVERSITY

Context-Aware Three-Stage Approach for Matching Resume to Job Descriptions

Master Of Science

in

Artificial Intelligence

September 18, 2023

By

Shubham Singh (21192874)

Dissertation Committee:

Dr. Amna Dridi

Dr. Edlira Vakaj

Faculty of Computing, Engineering and the Built Environment

Acknowledgments

I am deeply indebted to my supervisor, **Dr. Amna Dridi**, whose expertise, understanding, and patience added considerably to my graduate experience. I appreciate her vast knowledge and skills in areas related to my research and also for challenging me to navigate through issues independently. Dr. Amna Dridi was always accessible and gave constructive criticisms that improved the quality of this research work. Her responsiveness to emails, keen eye for detail, and rigorous reviews from the proposal stage to the completion of this thesis are highly valued. I feel privileged to have completed my journey at Birmingham City University under her guidance.

I would like to extend my heartfelt gratitude to my family for their love and moral support and providing me motivation along the journey.

Last but not least, thanks to my peers and colleagues who offered their encouragement and facilitated engaging discussions that enhanced my project experience.

ABSTRACT

In today's rapidly changing job market, job seekers often find it challenging to identify job postings that best align with their unique skillsets and experiences. The ubiquitous use of keyword-based matching systems, while efficient, tends to overlook the nuanced contexts that both job descriptions and resumes contain. To address this limitation, this report introduces a three-stage, context-based pipeline aimed at offering more accurate and personalized job recommendations to job seekers.

The first stage utilizes density-based unsupervised clustering techniques to categorize the available job descriptions in a database into contextually coherent clusters. This initial step enables a resume to be matched to a specific cluster of job descriptions, thereby narrowing down the range of jobs that align closely with the resume's content.

The second stage operates within the contextually matched cluster from Stage I and focuses on providing an overall contextual match between each job description and the resume. A cosine similarity measure quantifies this contextual matching, resulting in Stage II matching score.

In the third stage, attention shifts to a more granular level, focusing on the matching of specific skillsets extracted from both the resume and the job descriptions within the targeted cluster. Cosine similarity is used again to quantify this skill-based matching, generating a Stage III matching score.

Finally, an overall contextual matching score is computed by averaging the scores obtained from Stages II and III. Job descriptions with the top 10 highest overall matching scores are recommended to the job seeker, enhancing both the efficiency and accuracy of the job-matching process.

This research offers practical solution that could facilitate more effective job application decisions for job seekers.

Keywords: Job Market, Keyword-Based Matching, Context-Based Pipeline, Unsupervised Clustering, Cosine Similarity, Job Descriptions, Resumes, Skillset Matching, Contextual Matching Score.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Background and Rationale

The job search process has long been considered a challenging endeavor for job seekers. While the popularity of online job portals, such as LinkedIn and Indeed, have sought to address this complexity, the multitude of options available can still overwhelm candidates. These digital platforms provide an extensive repository of open job postings, allowing job seekers to explore specific positions with relative ease. Additionally, these portals streamline the application process, allowing individuals to track their applications and receive personalized job recommendations based on location and role preference.

Despite these advances, however, the abundance of opportunities paradoxically creates a new set of challenges. Job seekers are often faced with a bewildering range of choices, making it difficult to effectively narrow down their search parameters and identify opportunities that best match their qualifications, past experiences, and desired job attributes, including salary and career level (e.g., entry or senior positions).

Both freshers, who are entering the job market for the first time and experienced job seekers encounter challenges in effectively targeting their desired job applications from wide variety of available opportunities within the same role. Novice job seekers often lack the understanding of strategic techniques needed to target specific job positions, while seasoned professionals find it challenging to pinpoint the perfect roles that showcase their unique combination of skills and past expertise when dealing with generic online job applications. As a result, the job search process becomes a time-consuming endeavor for wide spectrum of job seekers, requiring hours of dedicated effort to navigate the multitude of job postings and to keep a close eye on job applications.

Hence, a potential solution to address this challenge, an automated solution could be developed to assist job seekers target a smaller and more focused set of job descriptions. This system would understand the context of requirements mentioned in online job postings and recommend jobs that closely match the context of job seeker's resume in terms of skills and past experience, therefore, reducing the likelihood of rejection. At the same time, such a system could assist recruiters in finding candidates who best match the requirements mentioned in their job advertisements. In this way, both job seekers and recruiters could potentially benefit from a more efficient and effective job-matching process.

There could be two solution approaches:

1. **Probabilistic Style Keyword-Based Matching Approach**: It is a primitive approach that involves determining the degree of matching the job descriptions based on common keywords present in both job descriptions and resume. Keywords are short words or phrases related to the requirements of a particular job [1]. The keyword-based matching approach can be illustrated below.

   Let $S_1,\ S_2,\ S_3,\ S_4,\ S_5\ ...\ \ S_r$ be the total skills as keywords extracted from a given resume of a candidate; $J_{11},\ J_{12},\ J_{13},\ J_{14},\ J_{15}\ \ldots\ \ J_m$ be the required skill-keywords from job description 1 and $J_{21},\ J_{22},\ J_{23},\ J_{24},\ J_{25}\ \ldots\ \ J_n$ be the required skill-keywords from job description 2. Hence, If some of skills from resume matches with job description 1, L_1 be the intersection of keywords common to both resume and Job Description 1, then,
   $L_1 = \{S_i\} \cap \{J_{1j}\}$ where,

   $$1 \leq i \leq r \text{ (r represents the total number of skills from the resume)}$$

   $$1 \leq j \leq m \text{ (m represents the total number of skills from job description 1)}$$

and L_2 be the intersection of keywords common to both resume and Job Description 2, then,

$L_2 = \{S_i\} \cap \{J_{2k}\}$ where,

1 $\leq i \leq r$ *(r represents the total number of skills from the resume)*

1 $\leq k \leq n$ *(n represents the total number of skills from job description 2)*

Then, Probability of Matching Skills for Job Description 1 (P_1):

$P_1 = \frac{L_1}{\Sigma S_i}$

And Probability of Matching Skills for Job Description 2 (P_2):

$P_2 = \frac{L_2}{\Sigma S_i}$

The probabilities P_1 and P_2 now indicate the likelihood of matching the skills mentioned in the job seeker's resume to the required skill-keywords in each job description, taking into account the intersection of skills between the resume and the respective job description.

Therefore, if P_2 is greater than P_1, then, it would mean that more skills in the resume align with the required skills in Job Description 2 compared to Job Description 1, indicating a better match between the resume and Job Description 2 in terms of only skills.

Hence, the job seeker can use these probabilities to assess the suitability of their resume for each job description and make an informed decision on which job description best aligns with their skill-set.

The limitation of the keyword-based approach is that it fails to effectively match candidates with job descriptions [2]. This is because it solely relies on the presence of specific keywords and has several extraction constraints, such as overlooking natural language semantics like synonyms, word combinations, and the contextual meaning of the content within the resume [3].

2. **Context Aware Matching Approach:** Context-aware job description to resume matching [4] is a sophisticated method that extends beyond simple keyword-based matching as this approach has the capability to understand relevance, relationships, and the context of words and phrases used. It utilizes the combination of advanced natural language processing and machine learning techniques to compare the semantics or context of resume and job descriptions rather than utilizing the rigid presence of keywords for comparison between resume and job descriptions to recommend a best matching job description among them. It could be illustrated using the following example:

Fictitious Resume Example	**Fictious Job Description Example**
Resume Summary:	Job Title: Machine Learning Engineer
I am a recent graduate with a strong background in computer science and a passion for machine learning. I have completed several projects using Python and TensorFlow, focusing on image recognition and natural language processing. I am eager to apply my skills in a dynamic team to drive innovation and solve real-world problems.	Description: We are looking for a talented Machine Learning Engineer to join our team. The ideal candidate should have a degree in computer science or a related field, with experience in Python and TensorFlow for developing machine learning models. Practical project experience in image recognition and natural language processing is a plus. We need someone who is enthusiastic about tackling challenging problems and contributing to cutting-edge research.

Figure 1.1: Illustration of Context Aware Matching Approach

While keyword-based approach could help in identifying the relevance between above given fictitious resume and job description by comparing the highlighted rigid keywords such as *"Python"*, *"machine learning"*, *"TensorFlow"*, *"image recognition"* and *"natural language processing"* from both. However, context-aware approach could help in comparing them by capturing the context and other nuances such as *"python"* and *"TensorFlow"* being used in context of *"machine learning"*, *"image recognition"* and *"natural language processing"* tasks and are associated with *"computer science degree"* or *"background in computer science"*.

Since, context-aware based approach extends beyond rigid keyword-based approach by capturing the semantics and context for comparison, it allows more flexibility to scale on huge volume of job descriptions to provide accurate and personalized best matching recommendations of job descriptions for a given resume.

One key limitation of this approach is that it is more computationally expensive than keyword-based approach, which only involves calculating probability as degree of job description matching by counting common keyword between in resume and available job descriptions. The increased computational requirements of the context-aware approach stem from its attempts to decode the inherent meaning, context, and relevance of keywords in the job descriptions and the resumes.

Regardless of the approaches described above, the fundamental objective is to identify a subset of job descriptions from a larger collection that most accurately aligns with the job seeker's resume. This alignment is quantified through a metric known as the **"matching score"** [5]. Consequently, both approaches aim to provide job seekers with a ranked list of job descriptions, ordered in descending sequence according to their respective matching scores. In both the keyword-based and context-based approaches, this score ranges from 0 to 1; the closer the score is to 1, the better the match between the job description and the job seeker's resume. While the keyword-based approach determines the matching score via the probability associated with common set of skill keywords present in each job description and in resume, the context-based approach calculates it based on the similarity score between each job description and resume by considering the context and semantics between them.

Hence, the primary motivation of this research is to design a three-stage context-based algorithm to recommend personalized jobs drawn from a diverse collection of job descriptions belonging to varied categories. **The recommendation is performed by ranking job descriptions in descending order according to a matching score determined by comparing each job description with the job seeker's resume**. Rather than using keyword-based matching, the context-based recommendation approach is used to match job descriptions and resume by capturing and comparing the semantic and contextual attributes from both, so that it could facilitate a more nuanced matching process based on job seeker's skills and experiences with requirements stated in job descriptions by the recruiter.

The rationale behind the three-stage context-based approach is to optimize the job matching process through structured, hierarchical filtering by systematically narrowing down diverse set of available job descriptions to recommend a small subset from it which best aligns with the context of job seeker's resume. Therefore, this research study explores different context-aware techniques to filter down job-descriptions in following three stages:

1. **Broad Filtering:** The first stage clusters or groups job descriptions into broader categories. This helps in reducing the search space by quickly identifying potential job categories that align with the applicant's resume. Instead of assessing a resume against every job description available, this initial broad categorization significantly narrows down potential matches.

2. **Focused Similarity:** In the second stage, each job description within the pre-identified clusters from the first stage is compared with the entire resume. This ensures a more detailed, context-rich comparison. By focusing only on clusters relevant from stage one, computational resources and time are saved, while also ensuring that matches made are contextually relevant.

3. **Skill-Level Analysis:** In the final stage, each job description within the identified cluster is further broken down into individual skills or requirements. These are then matched with the skills and experiences listed in the job seeker's resume. This granular comparison ensures that not only is the job role a good fit in general, but also that specific skill sets and experiences align closely with what the employer is seeking.

All these three stages could be combined in a pipeline to develop an automated and scalable solution that would output the matching score, as quantifiable measure of compatibility for top recommended job descriptions. By giving job seekers a ranked list based on the matching score, it would allow them to choose from the best options easily and quickly according to their own professional experience and skill-set, instead of being overwhelmed by countless job descriptions from their keyword-based job searches.

Chapter 2

Theoretical Background and Related Works

According to IBM [6], Natural Language Processing (NLP) is a branch of Artificial Intelligence that automates the mining and interpreting of text or human language to obtain valuable and actionable insights. This automation enables businesses to make fast decisions based on insights from large amounts of unstructured text data. Unstructured data, which makes up 80% to 90% of the information collected by businesses [7], doesn't follow a set pattern or structure. While it contains a wealth of information to guide business decisions, deriving insights from it using NLP has been a genuine challenge. In context of this research, the documents collected from online posted job descriptions is an example of unstructured text as they consist of free-form text that doesn't follow a consistent format or structure, making it more challenging to analyze and interpret systematically to find the best match contextually to resume of job seeker.

2.1 Evolution in Context Aware NLP Approaches

Earlier, NLP was guided by two main approaches: **Rule-Based Methods** and **Statistical Methods**, summary of which is tabulated below [8]:

Table 2.1: Comparison of Rule-Based Systems and Statistical Methods in NLP.
Source: [8]

Aspect	Rule-Based Systems	Statistical Methods
Definition	Based on predefined rules to analyze text and extract meaning.	Use statistical models to analyze and generate language, including language modeling and machine translation.
Challenges	Inability to handle ambiguity, context, and idiosyncratic language use; too rigid and inflexible.	Data sparsity and lack of context; struggle to capture complex relationships between words.
Why Outdated?	Limited by rigidity and inability to deal with real-world language complexity.	Although they addressed some limitations of rule-based systems, they still faced significant challenges in handling the complexity of natural language.
Contributions	Laid the foundation for NLP research and development.	Played an essential role in NLP evolution, paving the way for more advanced techniques like deep learning and transformers.

The inherent limitations of rule-based and statistical methods in failing to capture the context of words and phrases inside the text necessitated the development of more advanced NLP mechanisms using artificial neural network-based architectures such sequential-to-sequential learning mechanism [9] and attention-based mechanism [10] due to two primary reasons:

- To interpret the nuanced contextual characteristics or semantics of language found in text without the need for linguistic expertise.
- To extract valuable insights from vast volumes of unstructured text data, addressing the large-scale requirements of enterprises.

The method of sequential-to-sequential (seq2seq) learning [9], originally inspired by its application in machine translation, proves to be an effective approach to capture contextual information within text, given the inherent sequential arrangement of words. The Seq2Seq model comprises two primary components: **an Encoder and a Decoder**. These components are constructed using specialized cells like RNNs [11], LSTMs [12], or GRUs [13]. The Encoder accepts a series of words in a numerical format as input, converting them into an intermediary vector known as a context vector. Subsequently, the Decoder uses the context vector to predict the subsequent word in the sequence [14].

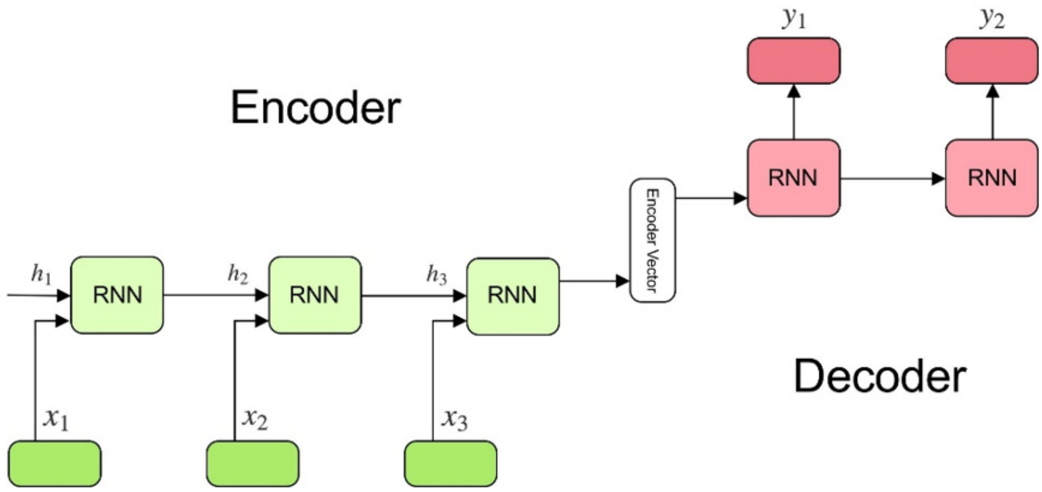


Figure 2.1: Encoder and Decoder in Se2Seq Architecture. *Source*: [14]

Using only the encoder component of Seq2Seq architecture, the encoding process transforms the sequence of words into a compressed fixed-size numerical format called context vector [15]. This numerical representation using context vector encapsulates the semantic and contextual information from a sequence of words in a text sentence. The Encoder's RNN, LSTM, or GRU [16] cells has the internal mechanism to maintain most or selective past information as memory from the preceding words, depending on the cell type [17] [18]. This ability allows the preservation of contextual information from the entire sequence of words, condensed into a fixed and compact context vector.

Due to auto-regressive property of this architecture, meaning, it predicts the next word by learning from past sequence of words, parallelization of learning process is the major limitation as the Encoder cannot process the sequence of words using parallel process. Another limitation stems from the property of recurrent memory cells – RNN / LSTM / GRU to maintain the long-term dependency in long word-sequences, which does not allow to preserve the extended contextual information in text containing lengthy information. [19]

Another limitation in Seq2Seq architecture arises from information bottlenecking between encoder and decoder due to single connection as context vector between them. This design compels all the encoded information produced by encoder from each word in a sequence to funnel through single context vector link, leading to loss of information from encoder to be decoded resulting in suboptimal quality in capturing context from sequence of words. This issue is compounded by the inherent limitations of recurrent memory cells, which struggle to retain information from long-sequence dependencies. [20]

To account for limitations mentioned earlier, a novel and state-of-the-art neural network architecture called Transformer, was proposed by Google in 2017, which also has encoder-decoder structure but instead based on self-attention mechanism to capture the relationship between words in a sequence [10]. As per proposed architecture in the paper, the encoder and decoder components of Transformer are composed of stack of 6 encoders and 6 decoders respectively and the output from last encoder is passed to all 6 decoders as shown in *Figure 2.2*.

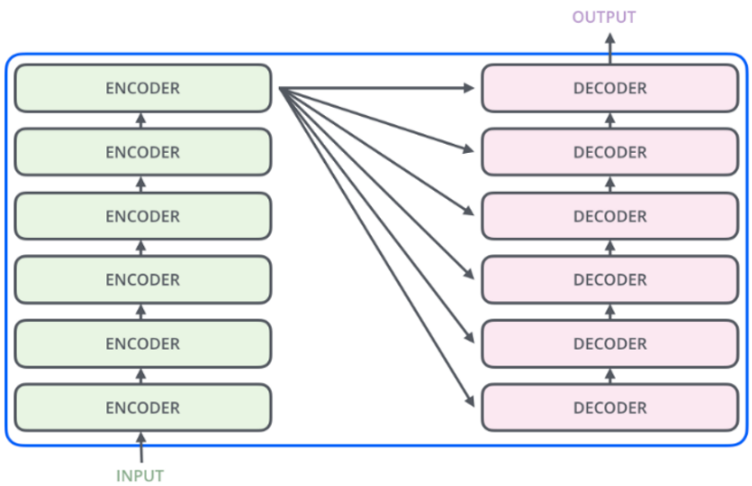


Figure 2.2: Transformers architecture. *Source*: [21]

The primary component in each encoder, which is composed of a self-attention layer and a feed-forward neural network, for capturing the context of a word from a sequence is the self-attention layer. It achieves this contextual understanding through the self-attention mechanism.

As per Google, self-attention is a mechanism in Transformer that aggregates information from all the other words in text to generate the attention score as a numerical representation of a word influenced by each word in the text to capture the entire context around that specific word [22]. Below example illustrates the working of self-attention with a simple text sentence. [23]

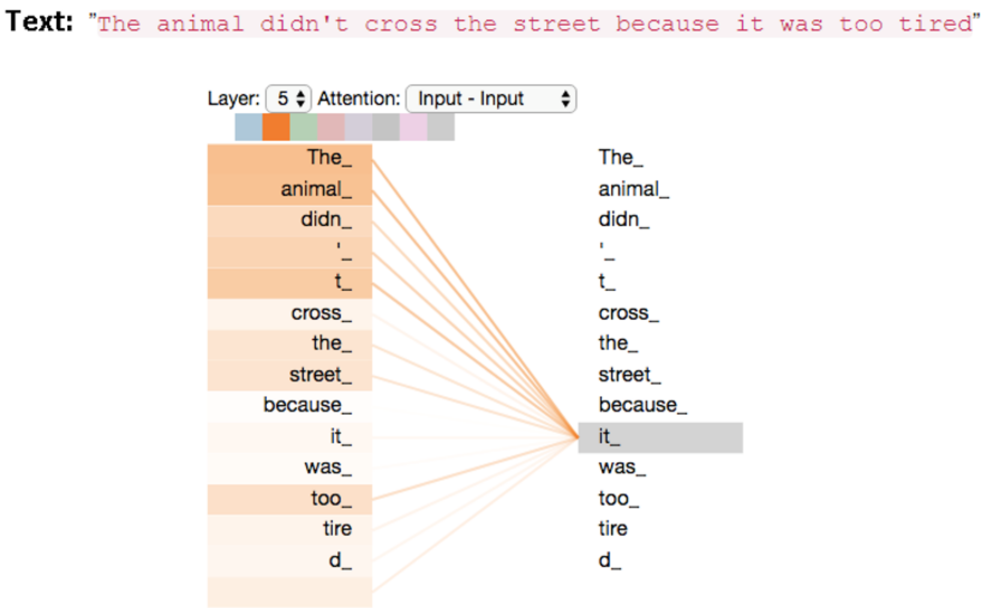


Figure 2.3: Attention Mechanism Illustration *Source*: [23]

As shown in *figure 2.3*, as the transformer processes the word *'it_'* from given text, self-attention has the ability to associate the context of word *'it_'* with word *'animal_'* as it assigns higher attention score (shown by darker color) to *'animal_'* while encoding the word *'it_'* after it aggregates the information from all the other words around and including word *'it_'* [23]. Hence, the transformer can encode the words considering the context from all the words around it in the text using self-attention mechanism. Instead of compressing all the information of a sequence into a single context vector, as in traditional seq2seq, the self-attention mechanism can weigh the importance of different words dynamically, ensuring that crucial information is retained. Also, since it does not utilize the recurrent memory cells like RNN / LSTM or GRU, it can preserve the contextual details from even longer dependencies in a sequence of words. Therefore, the transformer has the advantage of preserving the context from long-dependencies and the learning process can be parallelized from sequence of words [24]. [19]

Overall, Seq2Seq captures the context of a word from past sequence of words that come before it, while the Transformer relies on attention mechanism to capture a word's context from both its preceding and subsequent words within a given sentence of words.

The original Transformer architecture was primarily introduced to achieve state-of-the-art machine translation from English to German language through learned numerical representation encoded using self-attention mechanism from Encoder side and predicting the translated words from Decoder side. However, for other NLP related tasks that involve utilizing previously mentioned abilities of transformer, only the learned contextualized numerical representation of the text is often needed from Encoder component of the Transformer. Also, for documents containing multiple sentences, an additional change is required to account for separation between the sentences. Hence, Bidirectional Encoder Representations from Transformers (BERT) [25] was introduced in 2018, which is a pretrained transformer mainly composed of stack of only encoders (also, called Transformer Blocks) with self-attention mechanism within each encoder. The paper introduced BERT with two sizes – BERT Base, with 12 Transformer Blocks and BERT Large, with 24 Transformer Blocks.

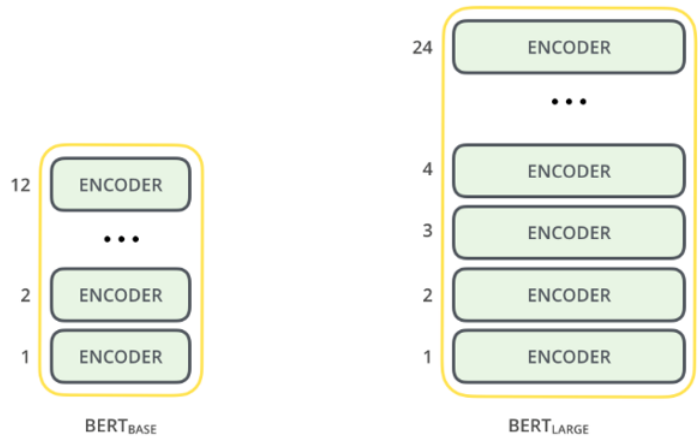


Figure 2.4: BERT Base and BERT Large *Source*: [23]

Moreover, BERT introduced additional special reserved tokens as positional cues, namely [CLS] and [SEP], for internal use in managing documents with multiple sentences. [CLS] is incorporated at the beginning of document to differentiate between distinct documents, while [SEP] is inserted between sentences to distinguish between sentences. Consequently, the entire document transforms into a sequence of tokens containing both these special tokens and individual words. This could be illustrated using a simple example below.

Document Example:

BERT is an amazing architecture. It can understand text very well.

Document as sequence of tokens:

[CLS], BERT, is, an, amazing, architecture, ., [SEP], It, can, understand, text, very, well, ., [SEP]

Using special tokens [CLS] and [SEP] for inter-document and inter-sentence separation respectively, BERT can be fine-tuned for multitude of NLP tasks such as text classification [26], semantic text similarity [27], question-answering [28] etc. [29]

Though BERT and other transformer models have set new standards in several natural language processing tasks, they still have limitations in generating sentence-level numerical representations. Transformers are designed for producing word or token-level embeddings, lacking a direct mechanism for sentence-specific numerical representations. Particularly in pair-wise NLP tasks such as Semantic Text Similarity (STS), which assesses the similarity between two sentences [30], where texts are grouped based on similarity, scalability becomes a concern when handling vast volumes of documents. This challenge arises from BERT's use in the cross-encoder architecture for STS task, visually represented in *figure 2.5*. Here, sentences A and B are ingested by BERT, processed together separated by a [SEP] token. A subsequent feedforward neural network then assigns a similarity score. However, the scalability issue becomes evident when considering a moderate dataset of 10,000 sentences. Such a scenario demands roughly 49,995,000 computations, translating to 65-hour inference time on contemporary GPUs. Using BERT for these tasks is impractical because of the high computational needs. [20]

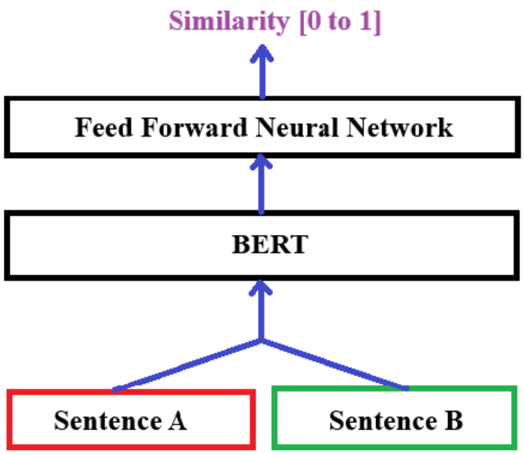


Figure 2.5: Semantic Text Similarity Task using BERT

To address the challenges of scalability and the need for efficient sentence-level numerical representations, Sentence-BERT (SBERT) was introduced in 2019 [31]. SBERT maintains performance levels comparable to BERT while significantly reducing inference time, especially for tasks like Semantic Text Similarity (STS). Utilizing Siamese Architecture [32], SBERT processes two input sentences using twin BERT models, as shown in *figure 2.6*. Each BERT model encodes the entire sequence of words or tokens in its respective sentence into numerical representations. To handle sequences of varying lengths, pooling strategies are used to derive a fixed-size numerical representation for each sentence. The outputs of these twin models are then combined (using methods like concatenation, subtraction, etc.) and fed through a feedforward neural network to produce a similarity score. The training objective of SBERT is to maximize this score for semantically similar sentence pairs while minimizing it for dissimilar ones. After training or fine-tuning, Sentence-BERT can produce fixed-length sentence-level numerical representations that capture the semantics and context of the sentences. Unlike the traditional BERT's cross-encoder approach, Sentence-BERT does not require pairing sentences with the [SEP] token to determine their similarity.

.

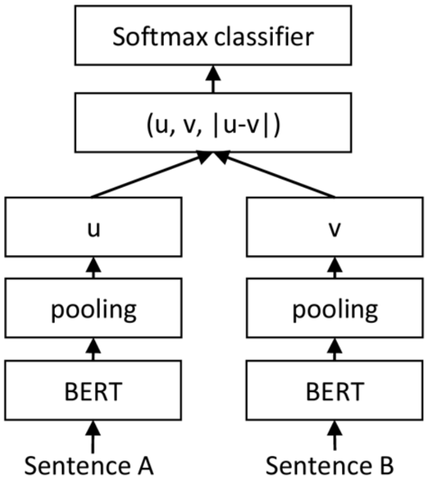


Figure 2.6: Internal Working Mechanism of Sentence-BERT. *Source:* [31]

In conclusion, the evolution of natural language processing methodologies has seen a significant progression from non-context aware to deeply context-sensitive approaches. Initially, rule-based and statistical methods provided foundational techniques for text processing but lacked the inherent ability to capture the nuanced contexts within which words and sentences operate. The seq2seq architecture was introduced as a big step forward as it could understand the context in short texts, but it had its flaws. The Transformer model came as an improved solution for capturing the context and semantics within longer text and was scalable due to its ability to parallelly process the sequence of words within text. Based on Transformer, BERT was introduced as encoder component from original architecture of Transformer, to capture the context of words in its numerical representation for other NLP tasks other than only machine translation but it was revealed that BERT was not suitable for text similarity and clustering task due to its high computation demand and high inference time. As a solution to this, Sentence-BERT achieved same state-of-the-art performance in capturing the semantics and context in its encoded numerical representation but was also optimized for performing scalable semantic text similarity using parallel twin BERT models as Siamese Network and encapsulating sentence-level context as numerical representation in its output.

2.2 NLP Encoding Techniques

To enable automated business decisions using the context from text, it's essential to convert the text into numerical representation that computer systems can process and interpret. These numerical representations of text are commonly known as vectors or embeddings. The technique used to extract features from the text in a numerical format supported by machine learning algorithms is known as **Encoding Technique / Embedding Technique or Feature Extraction** in the field of NLP.

Before any encoding technique or an advance NLP technique is applied on text, generally, the text is broken down into smaller chunks called token through the technique known as **Tokenization**, which essentially involves splitting the sentence or document into words or sub-words, making it easier for subsequent process easier to analyze and understand the text.

Considering an illustrative example using a simple sentence:

"Natural Language Processing is fascinating."

Without tokenization, computer system would consider this as a single entity, however, post-tokenization, it could be broken down into chunks called tokens as – [*"Natural"*, *"Language"*, *"Processing"*, *"is"*, *"fascinating"*]. Henceforth, each chunk can be individually vectorized using encoding techniques and processed ensuring that nuances and context within the sentence are effectively captured.

Stop words, such as *"is"*, *"and"*, *"the"* or *"it"* etc., are frequently occurring words that often provide limited semantic value when viewed outside their specific context and are often eliminated after tokenization. By removing these from a sentence, the remaining tokens, which typically carry more significant contextual meaning, become the focal point for vectorization using encoding techniques. This process helps in ensuring that the resulting vectors are less noisy and provide a more meaningful numerical representation of the text. In the given example, since, *"is"* is a common stop word and can be dropped resulting in more concise and contextually meaningful token list as [*"Natural"*, *"Language"*, *"Processing"*, *"fascinating"*].

2.2.1 Non-Context Based Encoding

Primitive encoding techniques often focus on the frequency or occurrence of each token in a text, without capturing the underlying semantics, context, or order of the tokens. One such technique is the Bag of Words (BoW) method [33], which represents text by counting the occurrences of each word but disregards the sequential arrangement of those words. In contrast, TF-IDF (Term Frequency-Inverse Document Frequency) encoding technique [34] not only considers the frequency of a word in a specific document (Term Frequency or TF) but also accounts for how common or rare the word is across an entire collection of documents or corpus (Inverse Document Frequency or IDF). The TF-IDF value is calculated by multiplying the TF and IDF. A high TF-IDF value indicates that the word is important in a given document relative to the entire corpus, while a low value might suggest the word is common across many documents and may not carry unique information for the given document.

In summary, both BoW and TF-IDF encode text based on the frequency or importance of specific words in a document, but they don't capture the semantic relationships between tokens. Moreover, they rely on a fixed vocabulary, which means words that are not present in the initial vocabulary (often referred to as 'out-of-vocabulary' or OOV words) aren't adequately represented or are simply ignored. Therefore, there's a need for encoding techniques that not only capture the semantic relationships between tokens but can also handle words outside the initial training vocabulary.

2.2.2 Context Based Encoding

To capture the semantic relationships between words, context-based encoding techniques have been developed. A prominent example is "word2vec," which uses a shallow two-layer neural network architecture to learn word representations by predicting words based on their surrounding context, or vice-versa [35]. There are two primary model architectures within word2vec:

1. **Continuous Bag Of Words (CBOW) Model:** This model predicts the target word from its surrounding context words.

2. **Skip-Gram Model:** Contrary to CBOW, this model predicts the surrounding context words from a given target word.

Word2Vec creates vector representations for individual words, capturing word-level semantics. While this is valuable for many NLP tasks, there are scenarios, such as Document Clustering or Document Similarity, where understanding the semantics of an entire document is crucial. In such cases, representing the entire document as a singular vector is more efficient and relevant. To address this need, Doc2Vec [35] was introduced, which can generate the fixed-length vector representation of text of any length like sentences, paragraphs or documents in a corpus using same word2vec CBOW mechanism internally.

Despite Word2Vec and Doc2vec's ability to generate the contextual vector representation of words or sentences / documents respectively, these models suffer from limited contextual awareness due to a parameter called context window that must be defined to generate vector representation. The context window [36] typically captures a fixed number of words around the target word. While it is great for understanding local context, it might still miss out on broader semantics that spans a larger portion of the text especially in long sentences or documents. Also, if the defined size is less, then, it would miss the broader context and if the size is too large, then it could introduce the noise while creating vectors.

Therefore, transformer-based models like BERT and Sentence-BERT addresses the challenges posed by word2vec and doc2vec by weighing the importance of each word in the entire document when generating the vectors, rather than relying on a fixed-size context window.

Given below is the tabulated comparative summary for all the Encoding techniques:

Encoding Technique	Description	Vector Size	Key Feature(s)
Bag of Words (BoW)	Represents text by counting the occurrences of each word, disregarding sequence.	Vocabulary size	Frequency-based, non-contextual.
TF-IDF	Encodes based on word's frequency in a document and its rarity across the corpus.	Vocabulary size	Weighs words by relevance.
word2vec	CBOW: Predicts target word from context. Skip-Gram: Predicts context from target word.	Typically 100-300 dimensions (but can vary)	Captures word-level semantics and relationships.
Doc2Vec	Generates a vector representation of entire documents or sentences.	Typically 100-300 dimensions (but can vary)	Captures document-level semantics.
BERT	Transformer-based model that weighs importance of each word in entire document using attention mechanism for vector	768 dimensions (base model) but can vary with model versions	Captures deep context throughout document.
Sentence-BERT	Transformer-based model specifically for sentences.	Typically same as BERT (e.g., 768 for base) but depends on model	Captures deep context of sentences.

Figure 2.7: Summary Of Different Encoding Techniques

2.3 Related Works

Online job platforms like Indeed and LinkedIn have modernized employment searches, offering features like resume-building and personalized job recommendations. However, the overwhelming number of listings can make the search daunting for job seekers, highlighting the inadequacy of traditional Information Retrieval (IR) methods [37]. This section examines recent advancements in using NLP to improve the efficacy of matching resumes to job descriptions.

Due to rapid changes in the labor market due to COVID-19 and digitization, the use of NLP to analyze and segment online job advertisements has become crucial as remote work opportunities expand and skill requirements evolve, both job seekers and employers face challenges in navigating the job market [38]. To address this, this comprehensive study conducted in 2023 examines specifically Lithuanian job market, using NLP and clustering techniques for segmenting job advertisements. As shown in *Figure 2.8*, The study outlines a three-step pipeline to cluster after cleaning and preparing extracted job advertisements:

1. Representing job profile requirements and skills into vectors using Embedding or Encoding Techniques.
2. Reducing vector size using Dimensionality Reduction Techniques.
3. Segmenting the job profiles using Clustering Techniques.

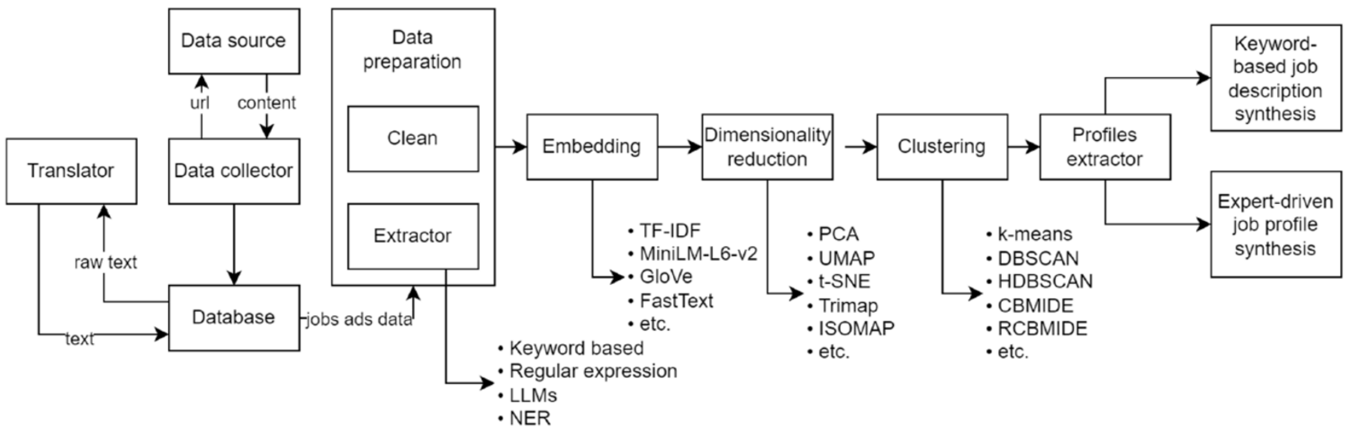


Figure 2.8: Job Profile Data Clustering Pipeline as described in Research Paper; *Source:* [38]

In the same study, various embedding techniques were evaluated for encoding job descriptions into numerical formats. These techniques included TF-IDF, Word2Vec, Doc2Vec, and BERT, out of which, BERT emerged as the most effective method for this purpose. The study also emphasized the need to reduce the vector size of the resulting job description vectors to eliminate noise and redundancy by using and assessing various dimensional reduction techniques such as PCA [39], UMAP [40], and t-SNE [41].

Furthermore, it highlights the challenge to maintain both global and local relationships when projecting high-dimensional job description vectors into lower dimensions. Out of these techniques, UMAP outperformed the others, being both computationally efficient and superior in terms of Trustworthiness metrics [42]. Finally, following previous two steps, it performed the comparative performance analysis of various clustering algorithms such as – K-Means [43], DBSCAN [44], HDBSCAN [45] etc., highlighting the appropriate selection of final optimized clustering algorithm specifically for segmenting job descriptions into optimal number of categories, while evaluating their performances using multiple clustering metrics such as silhouette coefficient [46], Davies–Bouldin index [47], and the Calinski–Harabasz index [48]. Among these clustering techniques, a density-based technique called HDBSCAN was experimentally found to be most effective as it automatically determines the optimal clusters on varying density distribution of vector-encoded job description in high dimensional space and due its robustness in handling noise data points.

In addition to the clustering-based approach for recommending job descriptions, another effective method for matching resumes with job opportunities involves feature extraction and vector representation as proposed in many previous recent studies [49] [50] [51]. Specifically, skills and other job requirements are extracted from both the resumes and job descriptions. These extracted features are then transformed into fixed-size vectors by aggregating the individual vectors of each skill and requirement present in both documents. The similarity between the resumes and job descriptions is then calculated using cosine similarity, which quantifies how closely the skills and requirements match. For example, in a recently published work, [52] used Word2Vec to transform the extracted skills, education, location and experience related keywords into individual vectors and aggregated using mean to represent both documents followed by determining cosine similarity to match and rank the suitable job for the job seeker. However, extraction of keywords requires parsing, which is not without information loss as suggested in the study [53]. This study performed resume parsing to match and rank their degree of suitability with the LinkedIn and non-LinkedIn job descriptions using BERT. Therefore, another study [54] directs towards the possibility of assessing the semantic relationship between whole resume and job descriptions by encoding them into embedding vectors using and comparing different encoding techniques without parsing keywords from job descriptions and resume.

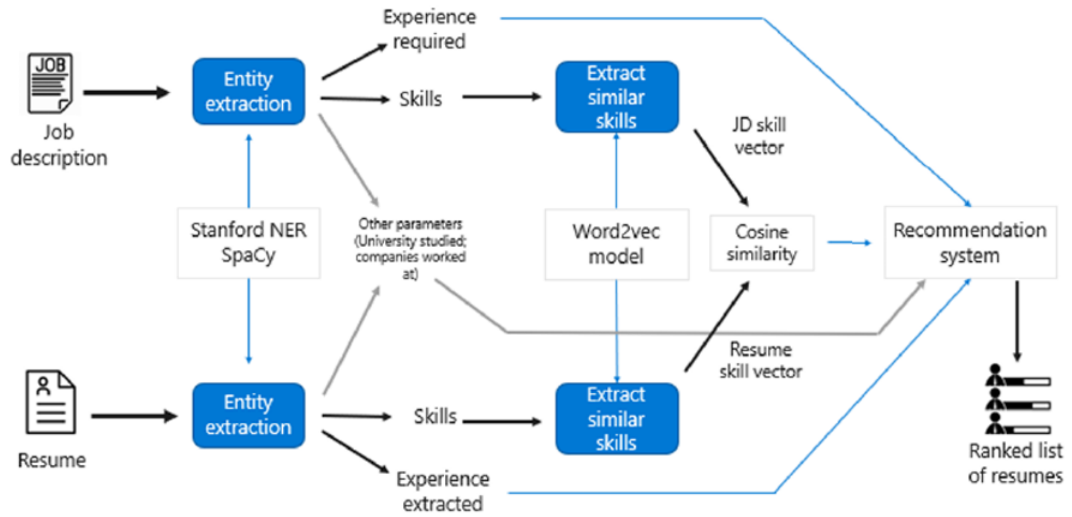


Figure 2.9: Resume and Job Description Matching using NER, Word2Vec and cosine similarity; *Source:* [55]

Although there is limited research focusing on skill-based resume and job description matching, one notable study from 2019 [55] used Named Entity Recognition (NER) to extract skill-related keywords from both resumes and job descriptions. These extracted skills were then vectorized using the Word2Vec model. Subsequently, cosine similarity was calculated between these skill vectors to produce a ranked list of resumes tailored to a specific job description as illustrated in proposed architecture given in the literature and shown in *figure 2.9*. While the 2019 study aimed at ranking resumes for a given job description, the focus of this thesis is the inverse: ranking job descriptions based on a given resume. However, a distinct approach was proposed by Nikita Sharma [56] [57]. This research introduced an innovative methodology for extracting pertinent skills from job descriptions, including emerging skills. The study applied Long Short-Term Memory (LSTM) neural networks [58] to facilitate this extraction. Furthermore, only noun-related entities were targeted for extraction, using the natural language processing library, SpaCy [59], and Part-of-Speech (POS) tagging techniques [60]. These extracted entities were subsequently classified as either "skill" or "not_skill" offering a nuanced and computationally efficient way to identify the most relevant skills for a given job description.

Building upon the insights and methodologies presented in mentioned previous research, this thesis aims to create a comprehensive approach to job and talent matching.

2.3.1 Research Objective and Contributions

This thesis focuses on developing an NLP pipeline to efficiently match job seekers' resumes with online job advertisements stored in the database. Building upon the related works described in the previous section, this work refines each of the three stages outlined earlier in later part of the *Section 1.1.* Rather than relying on keyword-based matching, this project employs context-based algorithms to match resumes with job descriptions. The following potential contributions are expected through the implementation of this pipeline:

1. **Personalized Context-based Resume to Job Descriptions Matching:** This research uses context-aware algorithms at each stage of the pipeline to capture the semantics and relationships between the requirements mentioned in both resumes and job descriptions. Hence, the solution provides personalized rankings and recommendations of job descriptions for job seekers, based specifically on context-based matching or similarity rather than rule-based or keyword-based matching.

2. **Deeper Semantic Level matching:** This research introduces a three-stage pipeline designed to perform contextual matching between resumes and job descriptions at increasingly specific levels. The first stage groups all job descriptions at the database level into contextually related clusters, making it easier to identify a suitable set for a given resume. In the second stage, the system captures the underlying context from each job description within the identified group. Finally, in the third stage, the pipeline assesses similarity at the skill-level to create a refined match with the given resume. Through this hierarchical approach, each

stage narrows the focus, providing a progressively more accurate ranking of job descriptions based on the context of the given resume.

3. **Efficient and Scalability:** The proposed pipeline methodology is designed to be modular, allowing for targeted optimization of encoding algorithms mentioned in *Section 2.2.2* at each stage of the process to capture contextual information from both resumes and job descriptions effectively. This design could potentially enhance the system's nuance, efficiency, and scalability to make accurate recommendations from a large volume of job descriptions, accommodating both LinkedIn and non-LinkedIn formats stored in the database.

Chapter 3

Methodology

3.1 Visual Representation of Research Methodology

As highlighted in later part of *Section 1.1*, below given figure provides an overview of the three-stage approach adopted in the research. In each stage, various context-aware techniques are applied to our collected set of job descriptions, details of which will be provided in subsequent sections.

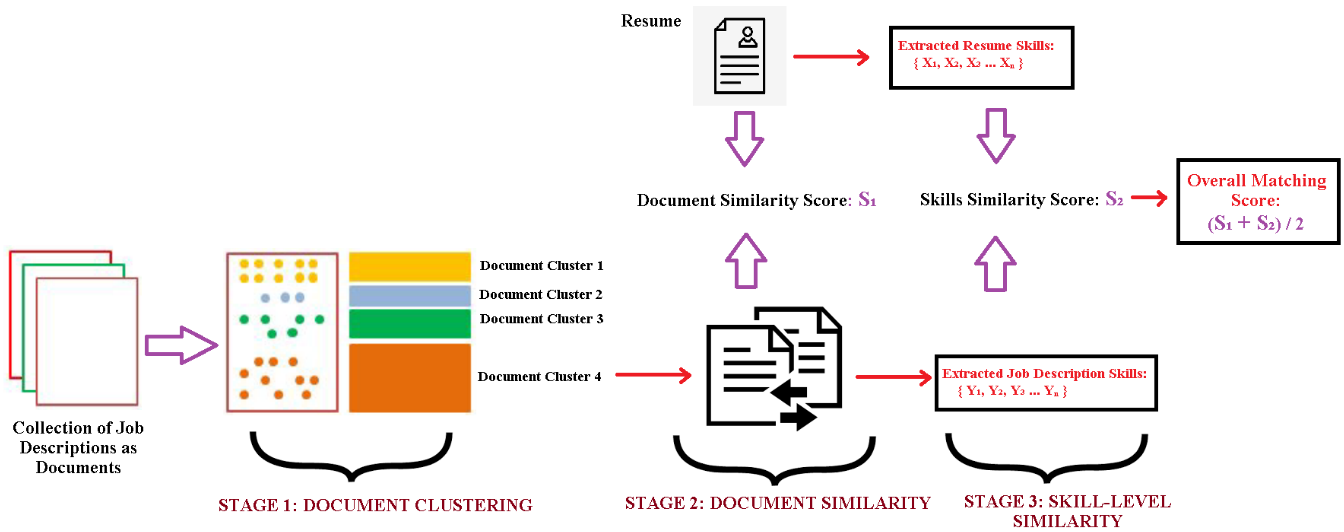


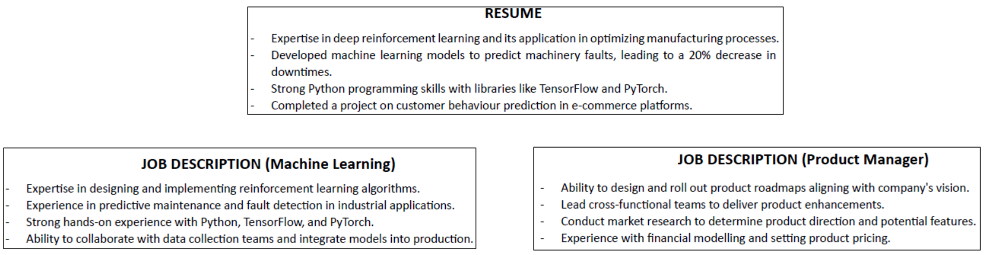
Figure 3.1: Pictorial Overview of the Methodology for Matching Resume with Job Description with Matching Score as an Output

Initially, the vast collection of job descriptions is grouped into an optimal number of distinct categories. These groupings are determined by the shared contextual and semantic meaning of the job descriptions within each category. The reasons for organizing these descriptions into clusters are as follows:

1. Most online job platforms store a large number of job listings in their databases, containing a broad variety of roles like software developer, machine learning specialist, product manager, and more. Directly comparing a resume with each of these individual listings can be computationally expensive and time-consuming. Therefore, it's more efficient to first sort these job descriptions into similar groups or clusters using an unsupervised learning technique.

2. After job descriptions receive cluster-based labels using a trained unsupervised learning model, the same trained unsupervised model can quickly and easily allocate a matching label to a resume, designating it to a particular category. In subsequent stages, the given resume of the job seeker will only be compared to job descriptions within its designated category. This streamlines the process, making the task of finding and recommending the best matching job descriptions much faster.

In the next stage, the resume is compared for contextual similarity with all job descriptions within the specific cluster it was assigned to in the earlier phase using the technique called Semantic Text Similarity (STS) [61] or Document-Similarity. The benefits for performing this stage can be illustrated using a hypothetical resume alongside two job descriptions: one for a machine learning role and another for a product manager, as detailed below:


RESUME
- Expertise in deep reinforcement learning and its application in optimizing manufacturing processes.
- Developed machine learning models to predict machinery faults, leading to a 20% decrease in downtimes.
- Strong Python programming skills with libraries like TensorFlow and PyTorch.
- Completed a project on customer behaviour prediction in e-commerce platforms.

JOB DESCRIPTION (Machine Learning)
- Expertise in designing and implementing reinforcement learning algorithms.
- Experience in predictive maintenance and fault detection in industrial applications.
- Strong hands-on experience with Python, TensorFlow, and PyTorch.
- Ability to collaborate with data collection teams and integrate models into production.

JOB DESCRIPTION (Product Manager)
- Ability to design and roll out product roadmaps aligning with company's vision.
- Lead cross-functional teams to deliver product enhancements.
- Conduct market research to determine product direction and potential features.
- Experience with financial modelling and setting product pricing.

Figure 3.2: Fictitious Resume and Two Job Descriptions (Machine Learning and Product Manager)

As elaborated in *Section 1.1*, the context-based matching system goes deeper than only keyword-based matching. From the fictitious example in *Figure 3.2*, the context surrounding terms such as *"optimizing manufacturing process"*, *"reinforcement learning"* and *"machine learning models"* from the resume demonstrates strong contextual parallels to phrases like *"reinforcement learning algorithms"*, *"predictive maintenance"* and *"fault detection in industrial applications"* present in the machine learning job description. This is not limited to direct keyword overlaps like *"python"*, *"TensorFlow"* or *"PyTorch"*. Contrastingly, the content from the product manager job description distinctly diverges in context from the fictitious resume. Thus, such document-level context-based matching can be extended to large set of documents as required by businesses for identifying semantically similar documents.

The final stage goes one level deeper into skill-level matching, where the list of skills extracted from all the job descriptions in the cluster determined in the first stage is contextually matched with the list of skills extracted from the given resume. Skill-level matching goes deeper into the content of both the job description and the resume. While document-level matching might conclude that a job description and a resume are similar based on shared broad context between them, skill-level matching assesses the depth of expertise based on specific skill-set in particular job role category. Instead of matching the exact skill-set present between resume and job description, this stage uses the semantic similarity of skill-set between them to ensure in-depth relevance of job seeker's resume with best recommended job description.

To illustrate skill-based semantic matching using the example illustrated in *Figure 3.2*, skills extracted from resume and both job descriptions are:

- **Skillset from Resume**: [*Deep Reinforcement Learning*, *Optimizing Manufacturing Processes*, *Predictive Maintenance*, *Python*, *TensorFlow*, *PyTorch*, *Customer Behavior Prediction*]
- **Skillset from Machine Learning Job Description**: [*Reinforcement Learning Algorithms*, *Predictive Maintenance*, *Fault Detection*, *Python*, *TensorFlow*, *PyTorch*, *Data Collection Integration*]
- **Skillset from Product Manager Job Description**: [*Product Roadmaps*, *Lead Cross-functional Teams*, *Market Research*, *Financial Modeling*, *Product Pricing*]

Upon manual semantic comparison of the above extracted skills from the resume with those from two job descriptions, summary table given below highlights a stronger alignment of skills between the resume and the Machine Learning Engineer position than with the Product Manager role:

Skills	Resume	Machine Learning Job	Product Manager Job
Deep/Reinforcement Learning	✓	✓ (Reinforcement Learning Algorithms)	-
Optimizing Manufacturing Processes	✓	-	-
Predictive Maintenance / Fault Detection	✓ (Predictive Maintenance)	✓	-
Python (TensorFlow, PyTorch)	✓	✓	-
Customer Behavior Prediction	✓	-	-
Data Collection Integration	-	✓	-
Product Roadmaps	-	-	✓
Leading Teams	-	-	✓ (Lead Cross-functional Teams)
Market Research	-	-	✓
Financial Modeling / Product Pricing	-	-	✓

✓ Represents **Skills present in Resume and semantically match Between Both Resume and Job Description**

✓ Represents Skills **NOT** present in Resume **but is related to Machine Learning Job Description**

✓ Represents Skills **NOT** present in Resume **but is related to Product Manager Job Description**

Figure 3.3: Simple Illustration showing semantically Skill-Set match Between Simple Fictitious Machine Learning related Resume and Two Job Descriptions each Related to Machine Learning and Product Manager

The degree of contextual or semantic similarity in Stages II and III is quantified using a matching score, based on cosine similarity [62]. **For each job description within the specific cluster assigned to the resume, an average matching score is computed for determining the overall matching score from Stages II and III. These average scores are then ranked in descending order, with the top 10 job descriptions being recommended for the respective resume.**

The subsequent sections delve into the technical components of the internal pipeline, including data cleaning, preprocessing, and the application of different context-based encoding techniques at each of the above-mentioned stages.

3.2 MongoDB: A Solution for Storing JD & Resume

To efficiently manage the storage and retrieval of both job descriptions and resumes, this research uses MongoDB as No-SQL database system [63]. This system is well-suited for handling and querying large volumes of unstructured data, making it ideal for text-based records like job descriptions and resumes. In the database (created as `'job-resume-db'`), job descriptions extracted from various sources are stored in one collection, referred to as `'job-descriptions'`, and the resumes in another, named `'resume'`.

The job description collection name `'job-descriptions'`, serves as the source for extracting relevant context from vast collection of job descriptions using a top-to-bottom, three-step hierarchical process, as detailed in an earlier *Section 3.1* and the resume collection name `'resume'` is used for matching the context of any given resume with the context from job description to recommend the best-matching job description. Therefore, by utilizing the MongoDB database, job seekers can match their respective resume from large pool of job descriptions in scalable, fast, and efficient manner on potentially business enterprise level.
[SEE APPENDIX FOR DATABASE AND COLLECTIONS]

3.3 STAGE 1: Document Clustering

Document clustering is an unsupervised learning method designed to categorize collections of text documents according to their inherent resemblances. The primary goal is to ensure that documents belonging to the same cluster exhibit greater contextual similarity to one another than to documents in distinct clusters. This means that clusters must be produced in a manner that maximizes contextual similarity within clusters (intra-cluster) while minimizing contextual similarity between different clusters (inter-cluster). [64]
[SEE APPENDIX FOR DOCUMENT CLUSTERING CODE]

3.3.1 Pipeline to obtain Optimal Number of Clusters

Any unsupervised learning methods typically cannot directly process data in its raw textual form, as discussed in *Section 2.2.* Therefore, each job description as a document text must be converted into numerical vectors that effectively capture their semantics or context for clustering. However, given many encoding techniques as shown in *Table 2.1*, the challenge lies in finding the appropriate encoding technique that is not only able to sufficiently able to capture the context from each job description as vectors but also enable the clusters to have maximum intra-cluster contextuality and minimum inter-cluster contextuality.

Another challenge arises when each job description is transformed into encoding vectors in a high-dimensional space. While this transformation aims to capture the context of each job description, it may introduce noise, as highlighted by the 'curse of dimensionality' [65]. This noise complicates the task of unsupervised learning techniques, making it difficult to learn patterns for creating optimal clusters of high quality [66]. To mitigate this, dimensions of the encoding vectors are often reduced before applying unsupervised learning techniques [66]. However, this dimensionality reduction is conducted carefully to ensure that the lower-dimensional space still captures the maximum context after transformation.

As the quality of the resulting clusters is highly dependent on the combined selection of encoding techniques, their parameters, and the parameters of the dimensionality reduction technique, each of these components is integrated into a single pipeline, as shown below in *Figure 3.4*, for determining both the optimal number of clusters and maintaining the quality of these clusters.

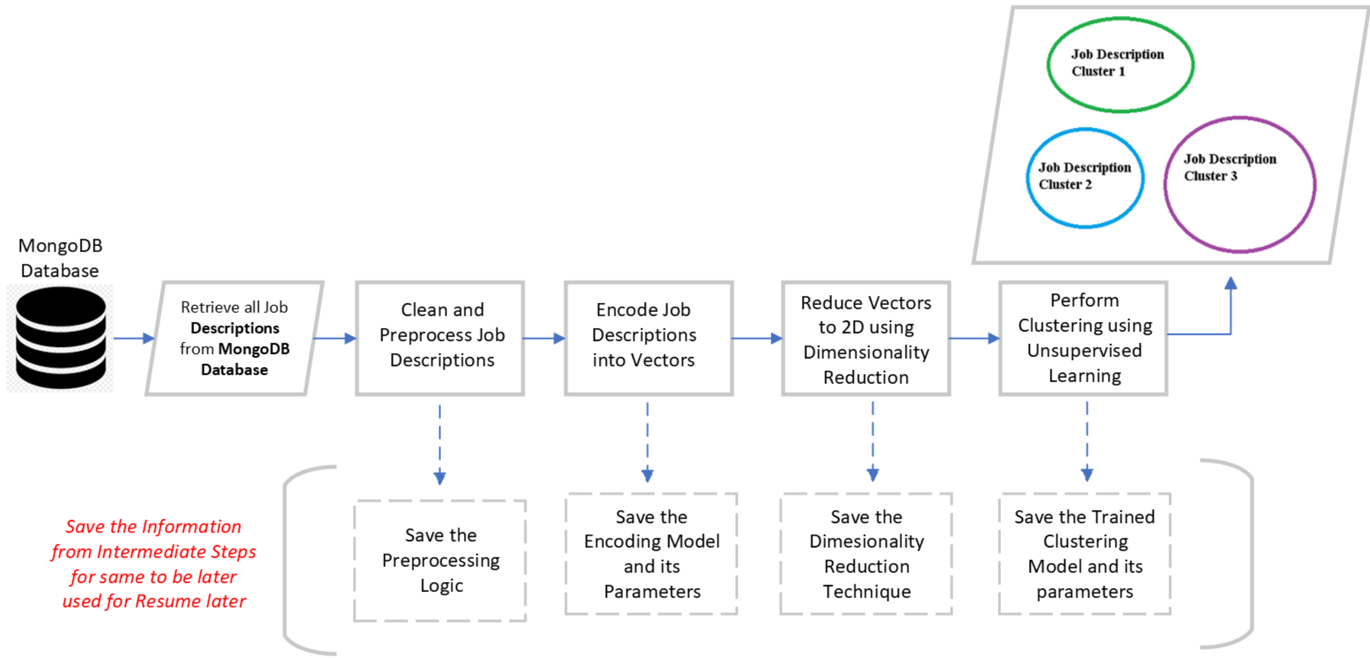


Figure 3.4: Clustering Pipeline Adopted in the Thesis

3.3.1.1 Document Preprocessing

A significant amount of job description data is unstructured and contains inconsistencies and noise. To make this data suitable for further analysis, it's necessary to clean, preprocess and standardize it. In this research, a preprocessing pipeline is designed specifically to enhance the consistency of each job description after retrieving them from MongoDB Database. As shown in the preprocessing pipeline *Figure 3.5* below, steps involve converting the text to lowercase, addressing inconsistencies by eliminating words with repeated letters, discarding all single letters except for 'c' (given its significance as a programming language), removing extra spaces, filtering out stop words (as detailed in *Section 2.2*), and keeping words primarily related to Nouns, Pronouns, and Adjectives through POS tagging [67]. [68]

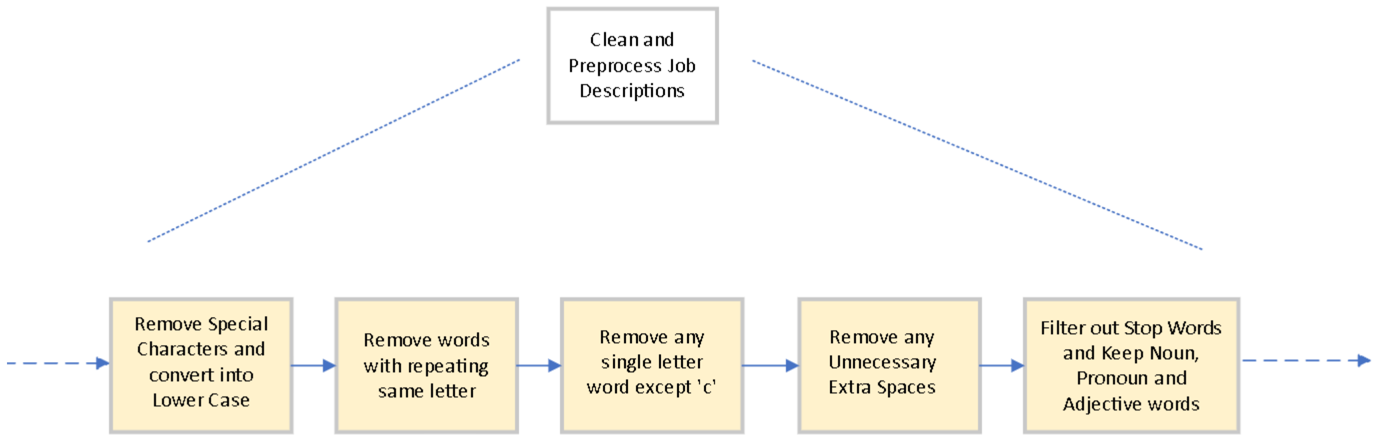


Figure 3.5: Clustering Preprocessing Pipeline Adopted in the Thesis

Since, noise can significantly affect the quality of cluster later [69] in this research, the above-shown preprocessing pipeline ensures that all job descriptions are free from significant noise and inconsistencies, while capturing main theme and maintaining context of each job description using **Nouns, Pronouns, and Adjectives**.

3.3.1.2 Document-Level Encoding Techniques

Among the many encoding techniques described in *Section 2.2*, only the context-aware based encoding techniques - **Word2Vec, Doc2Vec, and Sentence-BERT** - have been investigated in the research to encode the preprocessed job descriptions as documents into vectors or embedding. Because each of these encoding techniques has its own mechanism for capturing context of job descriptions into vectors, the challenge is not only to find the right set of parameters for these individual techniques, but also to find the best encoding technique capable of later producing high-quality clusters of job descriptions.

3.3.1.2.1 Word2Vec + TF-IDF As previously mentioned in *Section 2.2.2*, Word2Vec is used to create word-level vector representations within our job description corpus. However, there are key challenges associated with using Word2Vec for this purpose:

1. Word2Vec struggles with word ambiguity. Specifically, it generates the same vector for a word regardless of its context, leading to potential misinterpretations when the same word has different meanings in different documents. [70]

2. Since Word2Vec generates word-level embeddings, a certain mechanism is needed to aggregate these word-level vectors into a document-level vector. This is important not just for representing the job description document as a whole, but also for adjusting the vector representation of individual words based on their relative importance across documents, helping to mitigate the first challenge. This means that certain words that are more crucial to the job description's context could be given more weight or influence in the resulting document-level vector. Doing this can also help mitigate the issue of word ambiguity mentioned as the first challenge.

3. Another challenge is to determine two crucial parameters that influence the quality of the resulting word vectors: the context window size and the embedding size. The context window size determines how many words before and after a target word are considered as its context [71], impacting the quality of the contextual information captured. Meanwhile, the embedding size might potentially affect the resulting quality of the document-level vector representation, and consequently, the quality of the job description clusters.

As a solution to the above-mentioned points 1 and 2, the word embeddings could be aggregated to create document embeddings. Therefore, each Job Description document could be represented as a document embedding, denoted as:

$$X_i = \sum_{x_k \in A_i} w_{i,k}\, x_k.$$

Figure 3.6: Deriving Document Embeddings using Weighted Aggregation of Word2Vector Embeddings for each Word

X_i = weighted average of word vectors belonging to each job description ($x_k \in A_i$)

Where, w_i is the weight of each word vector $x_k \in A_i$, and A_i represents all the word vectors in the i-th job description.

The research, as mentioned in [72], proposes the possibility of choosing weights wi using the 'Term-Frequency-Inverse-Document-Frequency' (TF-IDF) approach, illustrated below:

$$w_{i,k} \equiv TF_{i,k} \times IDF_k$$

As a solution to above mentioned third challenge is to find the right balance for the set of parameters of Word2Vec using grid-search hyperparameter tuning [73], where different combinations of different values of the parameters of Word2Vec are experimented to create word embeddings aggregated and weighted using TF-IDF to get document embeddings to determine the optimal clusters of job descriptions based on clustering-related evaluation metrics, mentioned later in *Section 4.2.1*.

As per the official documentation of Word2Vec, below are parameters used in the research, for which a balanced combination of parameter values needs to be determined for optimal number of clusters using grid-search hyperparameter tuning. Additionally, the impact of each parameter on word embeddings and, consequently, on the clustering process is presented in the table below: [74] [75]

Table 3.1: Word2Vec Hyperparameters and their effect on Clusters

Hyperparameter	**Description**	**Effect on Clusters**
Vector Dimension	Number of dimensions in the word vectors.	Higher dimensions may capture semantic nuances, but overfitting is a concern.
Window Size	Size of the context window for context words.	Larger window captures more diverse context, potentially leading to broader but noisier word associations.
Min Count	Minimum frequency threshold for words to be considered.	Higher values exclude rare words, resulting in more general embeddings and potentially larger clusters.
Skip-gram (sg)	Algorithm choice: 0 for CBOW, 1 for Skip-gram.	CBOW (0) might perform better on frequent words, while Skip-gram (1) can handle rare words better.
Epochs	Number of times the model iterates over the dataset.	More epochs can lead to better convergence, capturing finer semantic nuances, but risk of overfitting.

3.3.1.2.2 Doc2Vec As mentioned in *Section 2.2.2*, Doc2Vec encapsulates the contextual information of job descriptions within document-level embeddings or vectors. Unlike Word2Vec, which directly generates word embeddings for each word from corpus of preprocessed job descriptions, the process of generating embeddings for job descriptions with Doc2Vec involves an additional set of steps. Following the preprocessing of job descriptions, the research uses the following five steps as inspired from [76] [77]

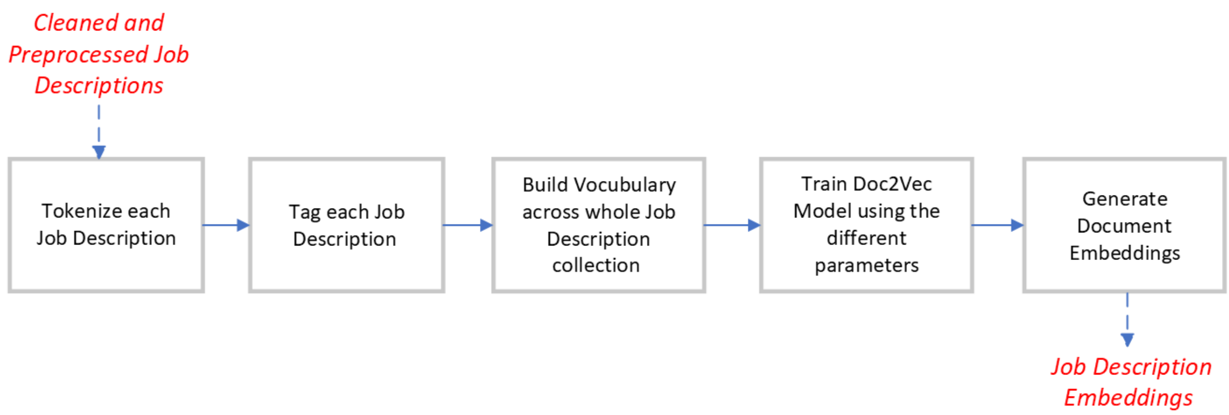


Figure 3.7: Steps taken for Doc2Vec Encoding after Cleaning and Preprocessing

- **Tokenize Documents:** Job descriptions are segmented into individual words or tokens.
- **Tag Documents:** A unique tag is assigned to each job description, enabling its identification.
- **Build Vocabulary:** A vocabulary is constructed from all the words present in the corpus of job descriptions.
- **Train Doc2Vec Model:** The Doc2Vec model is trained using tagged documents and vocabulary to learn embeddings.
- **Generate Document Embeddings:** The trained model generates embeddings that capture the semantic and contextual essence of job descriptions for clustering later.

As per the official Doc2Vec documentation [78], below parameters are experimented with different set of combination of values using grid-search hyperparameter optimization to determine quality number of clusters later:

Table 3.2: Doc2Vec Hyperparameters and their effect on Clusters

Hyperparameter	**Description**	**Effect on Clusters**
Vector Dimension	Number of dimensions in the document vectors.	Higher dimensions may capture more complex semantic relationships, but might also lead to overfitting.
Window Size	Maximum distance between the current and predicted words within a sentence.	Larger window captures broader context, which might lead to more generalized document embeddings.
Min Count	Minimum frequency threshold for words to be considered in the vocabulary.	Higher values exclude rare words, potentially impacting the richness of word diversity in document embeddings.
Epochs	Number of times the model iterates over the dataset.	More epochs can lead to better convergence, potentially improving the quality of learned document embeddings.

3.3.1.2.3 Sentence BERT As mentioned in *Section 2.2.2*, Sentence-BERT transforms each job description into a fixed-size embedding of 768 dimensions by capturing contextual information from both directions. As a pre-trained model, the research uses a specific version of its pre-existing architecture termed "all-mpnet-base-v2" to encode all job descriptions for clustering purposes. This choice of this specific architecture is due to the following factors: [79]

- **Diverse Pretraining:** The model's extensive pretraining on a broad and varied range of training data [80] makes it suitable for encoding each job description into a dense 768-dimensional vector. This capacity allows it to capture more nuanced contextual details from job descriptions, thereby enhancing the potential for generating high-quality clusters.

- **Pretrained Advantage:** Its pretraining eliminates the need for additional data preparation steps before inputting job descriptions, simplifying the encoding process and enhancing its applicability.

3.3.1.3 Dimensionality Reduction

In this research, to improve the clustering later due to aforementioned reasons in *Section 3.3.1*, an algorithm called Uniform Manifold Approximation and Projection (UMAP) [81] is used to project the high dimensional embeddings of each job description in two-dimensions. As mentioned in the given proposed literature for UMAP and highlighted in the comparative study [82], primary reasons for using it for dimensionality reduction on encoded job descriptions are:

Table 3.3: Reasons for Applying UMAP to Job Description Embedding

Reason	**Application to Job Description Embedding**
Non-linear Mapping	Job descriptions often contain intricate semantic relationships that linear methods like PCA might not capture well in lower dimensions. **UMAP's ability to handle non-linear mappings can help represent these complex relationships effectively in lower-dimensional space.**
Preservation of Local and Global Structure	In a high-dimensional embedding space, closely related job descriptions (in terms of required skills, roles, industries, etc.) would ideally be closer together, while dissimilar ones would be farther apart. **UMAP preserves these local and global relationships, making it easier to cluster or classify job descriptions after reduction.**
Computational Efficiency	Given that job description databases can be extensive, computational efficiency is important. **UMAP is faster than other non-linear methods like t-SNE, allowing for quicker processing of large datasets, making it easier to match job descriptions with resumes or analyze trends over time.**

3.3.1.4 Density-Based Clustering

As the final step in this clustering pipeline, unsupervised clustering algorithm called Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [83] is used to determine the optimal number of clusters from 2-dimensional vector representation of job description. It is a density-based unsupervised clustering technique, where clusters are seen as partitions of data with higher density compared to their surrounding areas, but it also accounts for noise, identifying and isolating data points with low density as outliers, rather than forcing them into existing clusters.

Table 3.4: Reasons for Applying HDBSCAN to Job Description Clustering

Reason	**Application to Job Description Clustering**
Handling Noise and Outliers	Job descriptions often vary widely in the quality and amount of information they provide. **HDBSCAN can identify and separate noise (or outliers) from clusters, which helps in grouping only the similar and relevant job descriptions together.**
Variable Density Clusters	Some clusters of job descriptions might be densely packed, indicating very similar roles, while others may be more spread out. **HDBSCAN can handle clusters of varying densities, making it ideal for capturing the nuanced differences between types of job roles.**
No Requirement for Predefined Number of Clusters	Traditional clustering algorithms like K-means require the number of clusters to be set in advance, which may not be ideal for job descriptions where the natural number of clusters is not known beforehand.**HDBSCAN does not have this limitation as it determines the number of clusters based on the data density.**

There are certain parameters of HDBSCAN, as shown in *Table 3.5*, whose values must be set on its initialization to determine the optimal number of clusters and their quality must be evaluated based on relevant evaluation metrics. In this research, **Relative Validity** [84], also called **Density Based Clustering Validation (DBCV)** [85] metric is an evaluation metrics used to assess the quality or "validity" of clusters produced by HDBSCAN while determining the optimal number of clusters. This metric is particularly well-suited for density-based clustering algorithms like HDBSCAN, as it evaluates clusters based on their compactness as well as separation. This makes it an ideal evaluation metric for clusters that can have varying shapes or forms, as highlighted in the referenced study [86].

To find the optimal set of parameter values for HDBSCAN using Relative Validity score, grid search hyperparameter tuning approach is used where different combinations of values of parameter is set to cluster the encoded job descriptions. Among the parameter combinations that yield a Relative Validity score **greater than 0.5** and **result in more than two valid clusters**, the one with the highest Relative Validity score is selected to determine the optimal number of clusters. According to the official HDBSCAN documentation [87], the table below provides a summary of key HDBSCAN hyperparameters used in this research, as these specific hyperparameters have a significant impact on the quality of the final clusters produced.

Table 3.5: Effect of Hyperparameters on Clustering

Hyperparameter	**Description**	**Effect on Clustering**
min_cluster_size	Specifies the minimum number of data points required to form a cluster.	Increasing this value will generally result in fewer, larger clusters. Lowering it could lead to more clusters, but they may contain noise or be less meaningful.
min_samples	Controls how conservative the clustering is by specifying the number of data points required in a cluster's neighborhood for it to be considered a core point.	Higher values make the clustering more conservative, focusing on denser core regions and potentially resulting in fewer clusters. Lower values may allow more liberal clustering with more clusters.
cluster_selection_epsilon	A distance threshold within which points are considered to belong to a cluster during the cluster selection process.	Increasing the value can merge clusters that are closer to each other, effectively reducing the number of clusters. Lowering it keeps clusters more distinct but may increase the number of clusters.

However, overall challenge in this whole job description clustering pipeline is that after cleaning and preprocessing all job descriptions (*Section 3.3.1.1*), combination of hyperparameters for each encoding technique, as mentioned in tables (*Table 3.1*) and (*Table 3.2*) and hyperparameters of HDBSCAN (*Table 3.5*) affects the quality of optimal number of clusters. Therefore, grid-search hyperparameter tuning is performed using combination of hyperparameters for one of the encoding techniques and hyperparameters for HDSCAN to determine the optimal number of clustering using Relative Validity as evaluation metrics, as illustrated in below pipeline.

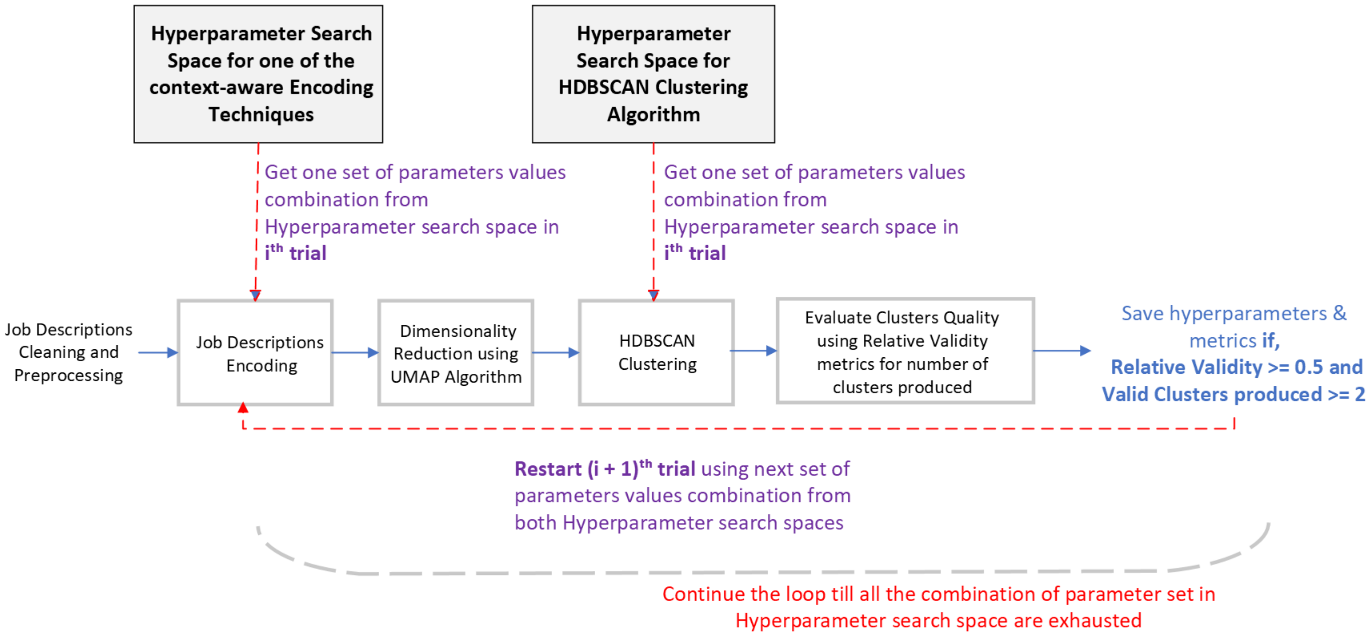


Figure 3.8: Grid Search Hyperparameter Tuning for Clustering Pipeline

As described pictorially above, in the grid-search optimization process, both Job Description Document-Level Encoding and the HDBSCAN clustering algorithm receive unique combinations of hyperparameters from a predefined search space. This includes a step for dimensionality reduction, reducing data to 2-Dimensions as an intermediate step in each trial. Following this, Relative Validity is used to assess cluster quality for the selected hyperparameters. **If Relative Validity exceeds 0.5 and produces more than two valid clusters, the results, along with the associated hyperparameters, are saved for future analysis**. This process helps in identifying the best optimal hyperparameter set for both document-level encoding and the HDBSCAN algorithm. Subsequent trials (i+1) continue in a similar manner with new combinations of hyperparameters.

3.3.1.5 Key Considerations for utilizing HDBSCAN Clustering

While experimenting with HDBSCAN, below mentioned key points are considered:

- Job Descriptions that could potentially be predicted as noise, generally represented using -1 as cluster data points after applying HDBSCAN, are removed from consideration.

- Since the cluster labels start at 0 and count up, the number of clusters is determined by finding the largest cluster label [88]. Hence, the actual number of clusters was found by increasing the largest cluster label by 1.

- To select the one final best encoding technique among Word2Vec + TF-IDF, Doc2Vec, and Sentence BERT, performance is evaluated and compared using K-Means clustering after previously identified optimal clusters through the density-based HDBSCAN algorithm, as K-Means requires a predetermined number of optimal clusters. The combination of two evaluation metrics—**Silhouette Score** [89] and **Davies-Bouldin Index** [47]—is used to guide this selection using K-Means, as described later in *Section 4.2.4*.

3.4 STAGE 2: Document Similarity

The second stage focuses on assessing the similarity between the job description and the resume, considering their overall context. The challenge in this stage lies in converting the text of the job description into document-level contextual embeddings or vectors, which most effectively captures the context of both using one of the three encoding techniques used in the research - Doc2Vec, Sentence BERT and BERT weighted with TF-IDF. This process involves a specific pipeline for preprocessing and cleaning the text, followed by training the document-level encoding technique on job description. The same pipeline containing preprocessing, cleaning, and document-level encoding steps will later be applied to the resume. However, on resume, only one of the trained document-level encodings will be used later in *Section 4.5* that performs the best in most effectively capturing the context from the job description after training.

[SEE APPENDIX FOR DOCUMENT SIMILARITY CODE]

3.4.1 Preprocessing and Cleaning to Remove Noise

To train the encoding techniques for capturing only the relevant context from the job descriptions, it is crucial to clean and preprocess them before encoding them into document vectors. Hence, the general cleaning procedure in the research follows the given steps below in *Figure 3.9*:

```
1. Input: Raw Text
   |
2. Remove keyword "title"
   |
3. Remove Special Characters
   |
4. Substitute Multiple Spaces with Single Space
   |
5. Remove Digits
   |
6. Apply Spacy NLP Model
   |
7. Filter Tokens Based on Conditions:
   |--> POS in [NOUN, PROPN]
   |--> Not a Stop Word
   |--> Not a Location (GPE)
   |--> Length > 1 or "C" or "R"
   |--> Not a Two-Letter Word with Repeating Letters
   |--> Not in List of Words to Remove (noisy_words)
   |
8. Output: Cleaned & Filtered Text
```

Figure 3.9: Preprocessing Steps Adopted in Thesis for Document Similarity

The preprocessing pipeline, shown in *Figure 3.9* for document similarity takes raw job description as input and first removes the keyword "title," and any special characters, followed by replacing multiple spaces with a single space and removing any digits if necessary. The text is processed through a SpaCy NLP Transformer model, called **en_core_web_trf** [90], after which the word tokens are filtered based on several conditions. Only tokens that are nouns or proper nouns, not stop words, and not geographical locations are retained. Additionally, tokens must either have a length greater than one except letters "C" or "R", due to their significance as programming languages and any two-letter words with repeating letters are removed. Furthermore, combination of manual and exploratory analysis is used to remove the unnecessary noise and less frequent keywords as they don't contribute to the meaning of the job descriptions and therefore, removing them helps in focusing the analysis on relevant context.

Table below summarizes the noise removal type and their descriptions:

Noise Removal Type	Description
Remove Noisy Words	Words that frequently occur in job descriptions but don't add value are removed to reduce noise. **Example:** [*experience, opportunity, description, requirement, applications*]
Remove 100 Least Common Words	Least frequent words are likely not important for the overall context. **Example:** [*insecurity, minority, secured, understudying, inaccuracy*]
Remove Bottom 100 Unigrams and Bigrams	The least frequently occurring unigrams and bigrams don't contribute significantly to the overall context. These combinations appear so rarely that they don't capture any generalizable features of job roles or qualifications and **their infrequent occurrence in the local vicinity of other words as local context means that they are not integral to the main themes or requirements in the job descriptions**. Therefore, they are removed to enhance the focus and quality of the analysis. **Unigram Example:** [*('mobilemarketingexecutive_job', 1), ('kb', 1), ('praccountexecutivepraccountmanager_job', 1)*] **Bigram Example:** [*('executive betting', 1), ('betting appointments', 1), ('appointments marketer', 1), ('marketer growth', 1)*]

Figure 3.10: Noise Removal Types adopted for Document Similarity

[SEE APPENDIX FOR LIST OF NOISY KEYWORDS, 100 LEAST COMMON WORDS, 100 UNI-GRAM AND BI-GRAM WORDS]

3.4.2 Measuring Effectiveness of Document Contextual Encoding

In this research, after the preprocessing and cleaning steps, the effectiveness of each contextual encoding technique in capturing the context from job descriptions is evaluated using a supervised approach. Hence, for evaluating this effectiveness, an additional feature of each job description, which is job category - 'product manager', 'software developer', 'chartered accountant' and 'machine learning' is used to assess how well the encoding captures the overall context of a job description effective enough to differentiate between these categories. The processed job descriptions, represented as contextually encoded vectors, serve as independent variables, while the job category, encoded as unique integer labels using Label Encoder [91], are the dependent variables. This encoded data is divided into **training** (70%), **validation** (20%), and **testing** (10%) datasets using stratified random sampling [92].

For training a machine learning model to differentiate between job categories from encoded job description contextual encodings using the training dataset, Support Vector Machine (SVM) model is used due to its effectiveness in classifying high-dimensional data [93]. For Doc2Vec Encoding, the model's performance is optimized using grid-search hyperparameter tuning on the test data, and its efficacy is evaluated using an unseen validation set. In the case of Sentence BERT and BERT model weighted with TF-IDF, hyperparameter tuning is not necessary since these models are already pretrained and their context capturing efficacy can be evaluated directly on validation dataset.

3.5 STAGE 3: Skillset Similarity

In the final stage, the degree of semantic similarity between the list of skills extracted from job descriptions and resume is determined using cosine similarity. Therefore, the initial challenge is to extract the set of skills from each job description in the database. Hence, in the research, skills are extracted using LSTM-based supervised learning approach, inspired from [94] and [95].

3.5.1 Skills extraction Pipeline for Job Description

As an overview of this approach, initially n-gram phrases are extracted using combination of Part-Of-Speech Tagging [96] and Regex Pattern [97]. The Regex patterns, shown in the *Figure 3.12*, are used to extract phrases and keywords mostly related to nouns and their variants using POS tagging. The extracted n-gram noun phrases are later manually labelled as 'skill' or 'no-skill' to supervise the training for LSTM-based neural network. Below pipeline illustrates this approach visually:

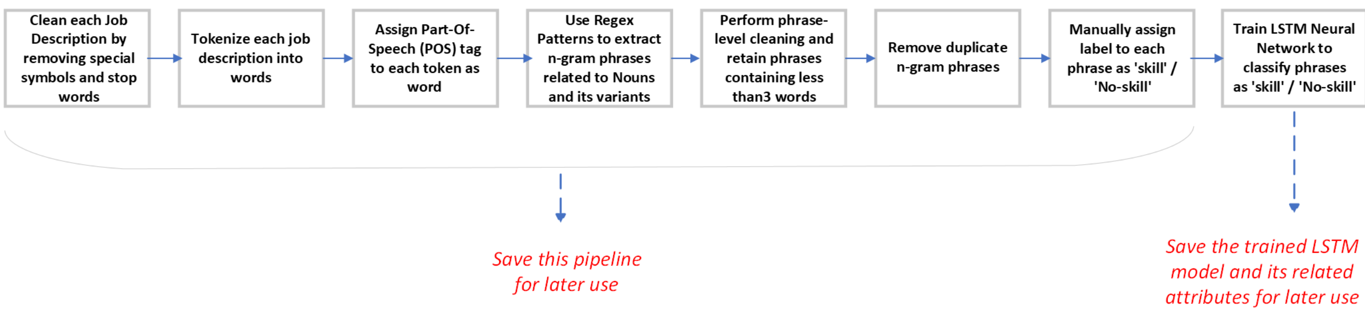


Figure 3.11: Pipeline for Extracting Phrases for Skills from Job Descriptions

[SEE APPENDIX FOR SKILL EXTRACTION PIPELINE CODE]

The Regex patterns used to extract the n-gram phrases are summarized in below table along with brief descriptions and examples of phrases extracted from job descriptions:

<table>
<tr><th>Pattern Type</th><th>Regex Pattern for POS Tagging</th><th>Description</th><th>Phrase Examples (along with respective POS tag) from Job Description after Regex</th></tr>
<tr><td>Noun Phrase Basic</td><td>{<DT>?<JJ>*<NN|NNS|NNP>+}</td><td>This pattern captures basic noun phrases that may start with a determiner (DT), followed by any number of adjectives (JJ), and ending with a singular noun, plural noun, or proper noun (NN, NNS, NNP).</td><td>[('software', 'NN'), ('developer', 'NN')]
[('graduates', 'NNS')]
[('engineers', 'NNS')]
[('machine', 'NN'), ('learning', 'NN')]
[('ai/artificial', 'JJ'), ('intelligence', 'NN')]</td></tr>
<tr><td>Noun Phrase Variation</td><td>{<IN>?<JJ|NN>*<NNS|NN>}</td><td>This pattern captures noun phrases that optionally start with a preposition or conjunction (IN), followed by any number of adjectives or nouns (JJ, NN), and ending with a singular or plural noun (NNS, NN).</td><td>[('big', 'JJ'), ('data', 'NN'), ('analytics', 'NNS')]
[('top', 'JJ'), ('tier', 'NN'), ('candidates', 'NNS')]
[('software', 'NN'), ('design/development', 'NN'), ('strong', 'JJ'), ('analytical', 'JJ'), ('programming', 'NN'), ('exceptional', 'JJ'), ('candidates', 'NNS')]
[('software', 'NN'), ('engineering', 'NN'), ('related', 'JJ'), ('academic', 'JJ'), ('background', 'NN')]</td></tr>
<tr><td>Verb Phrase</td><td>{<VBG|VBZ|VBP|VBD|VB|VBN><NNS|NN>*}</td><td>This pattern captures verb phrases that start with any form of a verb (VBG, VBZ, VBP, VBD, VB, VBN), followed optionally by a singular or plural noun (NNS, NN).</td><td>[('clustering', 'VBP'), ('technique', 'NN')]
[('experienced', 'VBD'), ('software', 'NN'), ('developer', 'NN')]
[('test', 'VB'), ('driven', 'NN'), ('development', 'NN')]</td></tr>
<tr><td>Nouns in between commas</td><td>{<NN|NNS>*<,><NN|NNS>*<,><NN|NNS>*}</td><td>This pattern captures series of nouns that are separated by commas. This is particularly useful for extracting lists of skills or requirements.</td><td>[('framework', 'NN'), (',', ','), ('software', 'NN'), ('developer', 'NN'), (',', ','), ('programmer', 'NN')]
[('developer', 'NN'), (',', ','), ('web', 'NN'), ('designer', 'NN'), (',', ',')]
[('backend', 'NN'), ('developer', 'NN'), (',', ','), ('php', 'NN'), (',', ','), ('software', 'NN'), ('architect', 'NN')]</td></tr>
</table>

Figure 3.12: The Regex Patterns used to Extract the N-Gram Phrases

This straightforward yet advanced pipeline method enables scalable extraction of skill requirements from job descriptions. While it can be extended for business-level applications, a drawback is the initial time investment required for manually labeling the extracted phrases. However, once the model is trained, it can rapidly identify skills and other requirements in any job description.

Once the LSTM neural network is trained, the phrase extraction pipeline and the trained LSTM model, along with its associated attributes, can be used later to quickly identify skill requirements from job descriptions in a specific cluster in which job seeker's resume is predicted to belong after the first stage.

3.6 Resume To Job Description Matching

After processing the job descriptions using the three-stage pipeline described earlier, all crucial components— including preprocessing logic, trained encoding models, clustering algorithms, and the LSTM model—are saved for future use. These learnt and saved attributes from the previous 3-stage context extraction pipeline from job descriptions allows the contextualization of job applicants' resumes in same way. However, the final pipeline only uses only one trained contextual encoding / embedding technique in each stage which most effectively extracts context from job description, so that only same best performing encoding technique can be used for extracting context from resume to match both effectively and accurately in each stage.

This section describes the methodology for integrating resumes into the previously described trained pipeline, with the objective of matching the contextual information in resumes with that in job descriptions. The degree of this contextual match is quantified using cosine similarity, as derived from stages II and III of the pipeline. Specifically, the average of the matching scores—based on overall context from stage II and specific skills from stage III—is used as the final matching score. This score is then used to recommend the top 10 job descriptions that most closely match with the given context and skills in the resume.

[SEE APPENDIX FOR RESUME MATCHING TO JOB DESCRIPTION PIPELINE CODE]

3.6.1 Cluster Label Assignment to a Resume

Initially, each resume is assigned a cluster label as one of the optimal number of clusters previously determined, as discussed in *Section 3.3.1.4*. This helps to identify the cluster of job descriptions to which the resume contextually belongs. To achieve this, text information for a specific resume is retrieved from MongoDB using its resume ID. The contextual information from resume text is then encoded into high-dimensional vectors or embeddings using the encoding technique that most effectively captured the job description context earlier in Stage I, as outlined in *Section 3.3*. These high-dimensional vectors are subsequently reduced to 2D using the same UMAP algorithm employed earlier. Finally, K-means algorithm, trained earlier to group job description contextually, is used to predict or assign the appropriate cluster label for the resume. This entire process is illustrated in the pipeline below:

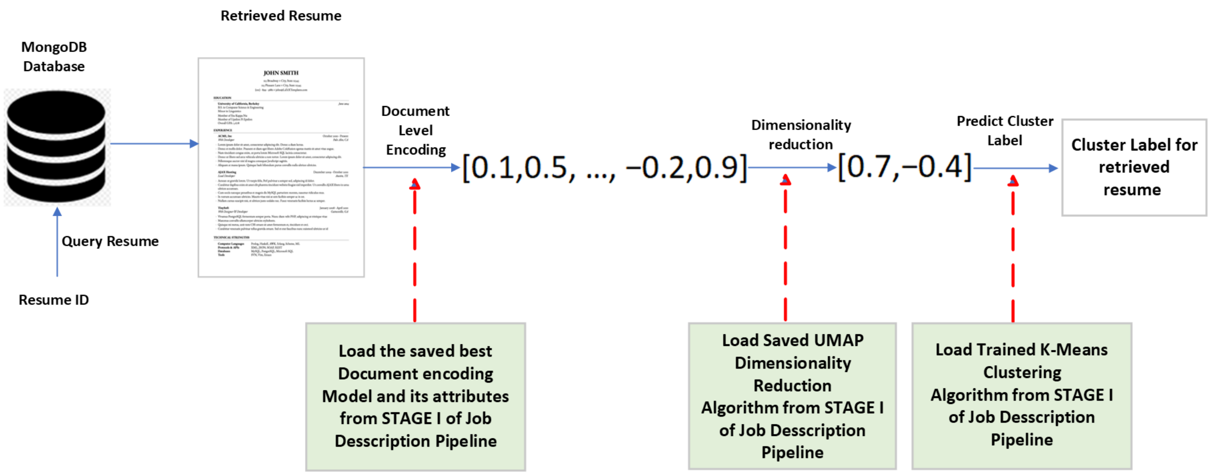


Figure 3.13: Cluster Label Assignment to a given Resume

3.6.2 Match Overall Resume to Job Description in a Cluster

Next, a matching score, which represents the contextual and semantic similarity between a given resume and all job descriptions retrieved from MongoDB is calculated. These job descriptions are specifically from the cluster to which the resume was previously assigned. To ensure consistency, the same preprocessing and cleaning steps described in *Section 3.4.1* are applied to both the resume and job descriptions to remove irrelevant noise and rare words from both. The document-level encoding technique, which proved most effective at capturing context in job descriptions in stage II (as discussed in *Section 3.4.2*), is used to encode both the resume and job descriptions. Finally, the cosine similarity metric is used to calculate the matching score between the resulting resume and job description embeddings. This matching score measures the overall contextual similarity between resume and each job description. The scores are then sorted in descending order, with each score being associated with the respective job description ID for easy reference later as illustrated in pipeline below:

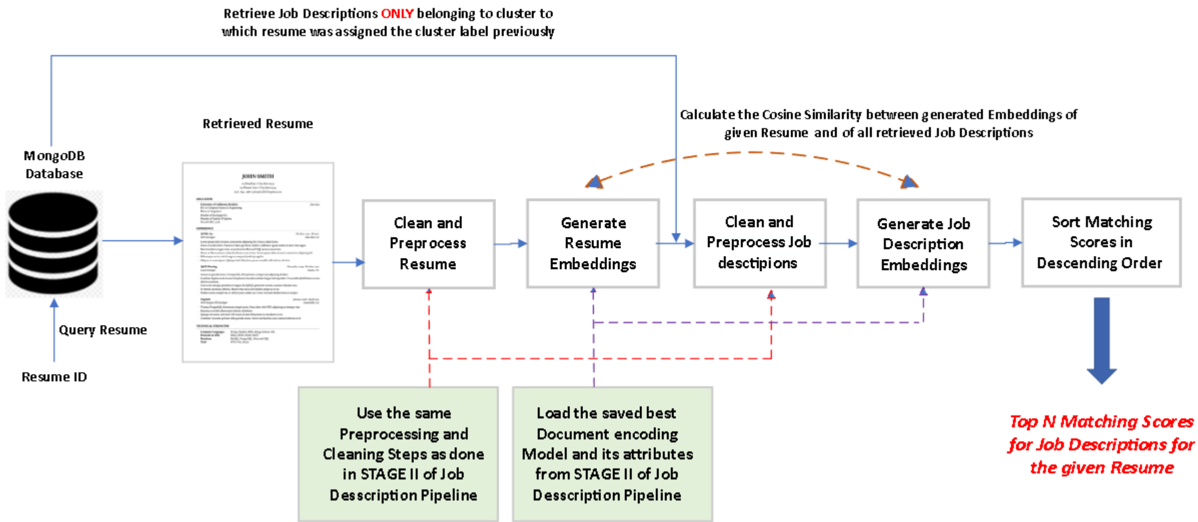


Figure 3.14: Pipeline for Matching Overall Resume to Job Descriptions in a specific Cluster.

3.6.3 Match Skillset in Resume with Job Description in a Cluster

Finally, a semantic matching score is calculated based on the similarity of skill sets extracted from both the resume and the job description within the same cluster. Skills are extracted from resumes using the Python library ResumeParser [98], while skills and other job requirements are obtained from the job descriptions using the trained skill extraction pipeline, tokenizer, and LSTM model described in Stage III (see *Section 3.5.1*). Each extracted skill from both the resume and the job description is vectorized using BERT and its associated tokenizer. Pairwise cosine similarity [99] is computed between each skill set from the resume and each skill set from the job description. These pairwise scores are then averaged to obtain a final matching score. These final matching scores are subsequently sorted in descending order by their associated job description IDs for easier reference later, as illustrated in the pipeline below.

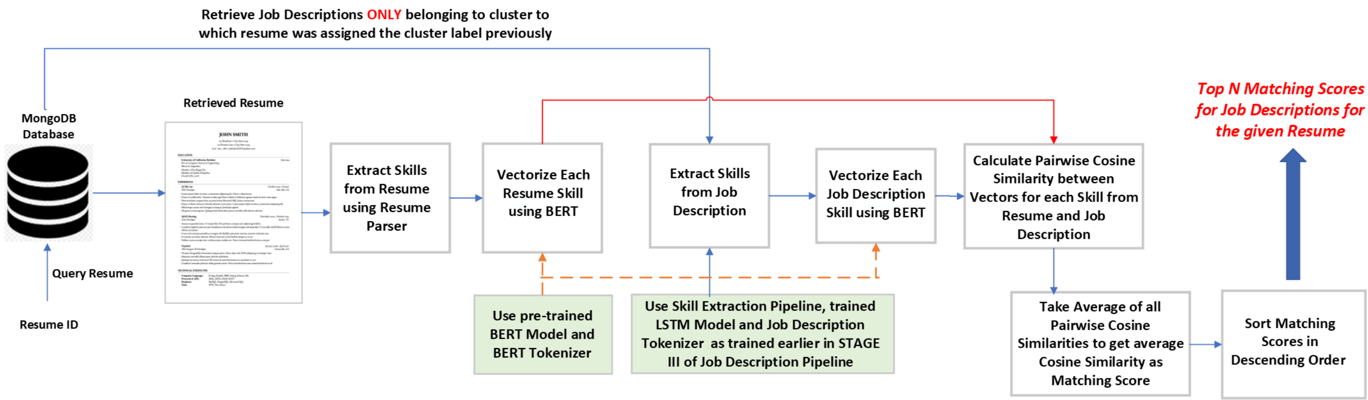


Figure 3.15: Pipeline for Matching Skillset from Resume to Skillset from Job Descriptions in a specific Cluster.

3.6.4 Recommending Best Job Description for a Given Resume

Once matching scores are calculated for each job description based on both overall context and skill set, an average matching score is computed for each job description ID. These average scores are then sorted in descending order to identify the top 10 job descriptions that best match a given resume.

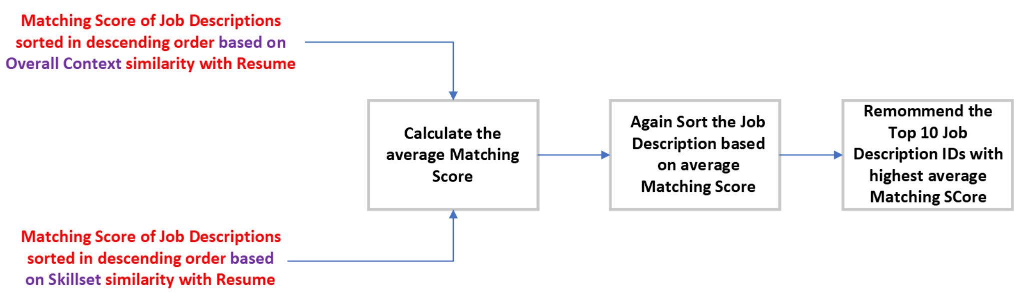


Figure 3.16: Recommending Best Job Description for a Given Resume with Matching Score

3.7 Potential Limitations

Despite a comprehensive methodology presented in this chapter, there are some limitations to be considered as mentioned below:

- The proposed methodology is limited to job descriptions and resumes written in English, which may restrict its applicability to other languages and diverse job markets.
- Due to the heavy computational demands of context-aware algorithms, the job description dataset stored in MongoDB is limited to four main related categories: **Machine Learning, Software Developer, Chartered Accountant, and Product Manager.**
- Potential biases may arise from optimizing encoding techniques to capture context from job descriptions, particularly when there are unequal numbers of job descriptions in each of the categories previously mentioned.

Chapter 4

Experiments and Result Discussion

4.1 Dataset

To ensure a comprehensive dataset that reflects the diverse range of job descriptions across various roles, this thesis utilizes data from two distinct sources: 1) A Kaggle dataset [100], which includes job titles and corresponding descriptions from established online job portals such as CV-Library [101] and Totaljobs [102], and 2) Manually scraped job descriptions from LinkedIn Jobs.

Due to computational limitations, this research narrows its focus to four job categories that are in high demand, as recommended by Google. These categories are **Machine Learning, Software Development, Chartered Accountancy, and Product Management**. Job descriptions corresponding to these categories are selectively filtered from the Kaggle dataset and supplemented with additional descriptions sourced manually from LinkedIn job advertisements.

[SEE APPENDIX FOR JOB DESCRIPTION DATASET PREPARATION AND EXTRACTION FOR MENTIONED CATEGORIES CODE]

[SEE APPENDIX FOR RESUME DATASET PREPARATION AND EXTRACTION FOR MENTIONED CATEGORIES CODE]

The distribution of job descriptions in each category is as follows:

- **Chartered Accountant:** 562
- **Machine Learning:** 161
- **Product Manager:** 411
- **Software Developer:** 2398

In addition to job descriptions, publicly available resumes are also collected for each category. These are sourced either from online project portfolio blogs or from LinkedIn profiles where job seekers have made their resumes publicly available to recruiters.

The breakdown of collected resumes across the selected categories is:

- **Chartered Accountant:** 5
- **Machine Learning:** 6
- **Product Manager:** 1
- **Software Developer:** 9

By narrowing the scope to these four job categories, this research aims for a manageable yet diverse dataset that allows for robust analysis, while also being mindful of computational constraints.

4.2 JD Clustering Experiments and Result

To optimize the clustering of job descriptions in database, a series of experiments were conducted that involves a combination of encoding techniques, dimensionality reduction methods, and the HDBSCAN clustering algorithm. Various hyperparameters for encoding techniques and HDBSCAN were fine-tuned to achieve optimal clustering, the evaluations and impact of which are summarized in subsequent sections.

4.2.1 Hyperparameter Tuning Result for Word2Vec + TF-IDF and HDBSCAN

Given below are the ranges of hyperparameters that were used for both Word2Vec and HDBSCAN algorithms.

Table 4.1: Range of Hyperparameters for Word2Vec and HDBSCAN Clustering

Word2Vec Parameters	Range
Vector Size	50, 100, 200
Window Size	2, 5, 10
Minimum Count	2, 5, 10
CBOW/Skip-gram	0 (CBOW), 1 (Skip-gram)
HDBSCAN Parameters	**Range**
Min Cluster Size	Computed (Range: $1, \ldots, 2\log n$) ; where n is Embedding size
Min Samples	Computed (Range: $1, \ldots, 2\log n$) ; where n is Embedding size
Cluster Selection Epsilon	0.0, 0.1, 0.2, 0.4, 0.5, 0.8, 1.0

Based on each combination of hyperparameters, qualitative and quantitative evaluation summary is given below to determine the best hyperparameters for Word2Vec and HDBSCAN:

4.2.1.1 Overall Validity Score Vs CBOW and Skip-Gram Type

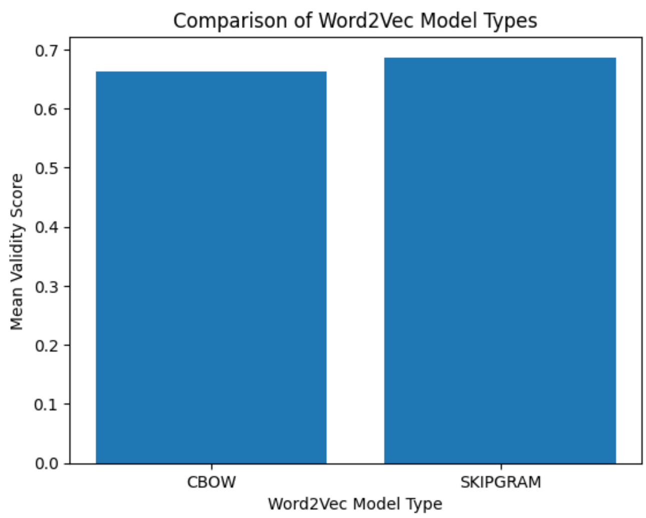


Figure 4.1: Comparison of Word2Vec Encoding types (CBOW and Skip-Gram) based on Overall Mean Validity Score

As illustrated in the above comparison figure, the Skip-Gram variant of the Word2Vec encoding model demonstrated only a marginal performance advantage over its CBOW counterpart. As described in *Section 3.3.1.4*, the Validity Score serves as an indicator of the quality of the clusters generated. Therefore, the similar Overall Mean Validity Scores suggest that both CBOW and Skip-Gram models produce clusters of job descriptions that are nearly equivalent in quality.

4.2.1.2 Validity Scores Vs Embedding Size / Vector Size for Each Type

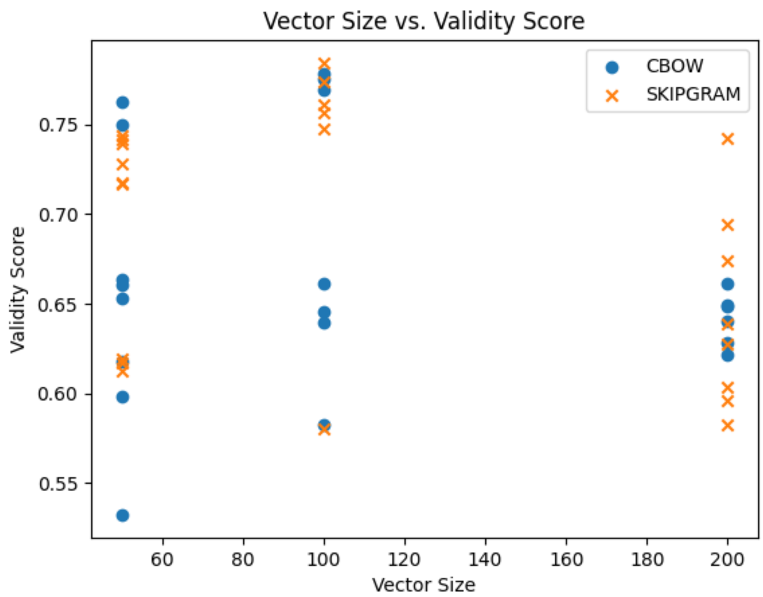


Figure 4.2: Validity Scores for different Vector Sizes of Each Word2Vec Type Encoding Algorithms

In the above graph, the clustering quality and performance of CBOW and Skip-Gram varies with different vector sizes. Specifically, CBOW achieves its best performance of maximum validity score with a vector size of 50, while Skip-Gram achieves it with a vector size of 200. Interestingly, the difference in maximum validity scores between Skip-Gram and CBOW is minimal when the vector size is set to 100. The peak overall maximum validity score was obtained with a vector size of 100, suggesting that this might be the optimal vector size for encoding job descriptions.

4.2.1.3 Overall Mean Validity Scores for Number of Clusters Produced (Validity score Aggregated per cluster)

Figure 4.3: Overall Mean Validity Score for Number of Clusters produced by each Word2Vec Type

As indicated by the above graph, the **Continuous Bag-of-Words (CBOW) model achieves its highest average validity score when producing five clusters**. In contrast, the **Skip-Gram model reaches its peak mean validity score with four clusters**. **The above shown average validity score is aggregated on each Word2Vec model type within each of the above shown clusters produced**. This suggests that number of Optimal clusters for given categories of Job Descriptions could be either 4 or 5, which is expected because there has been four actual job categories present in the research as mentioned in *Section 4.1*

4.2.1.4 Validity Score for different Cluster Sizes

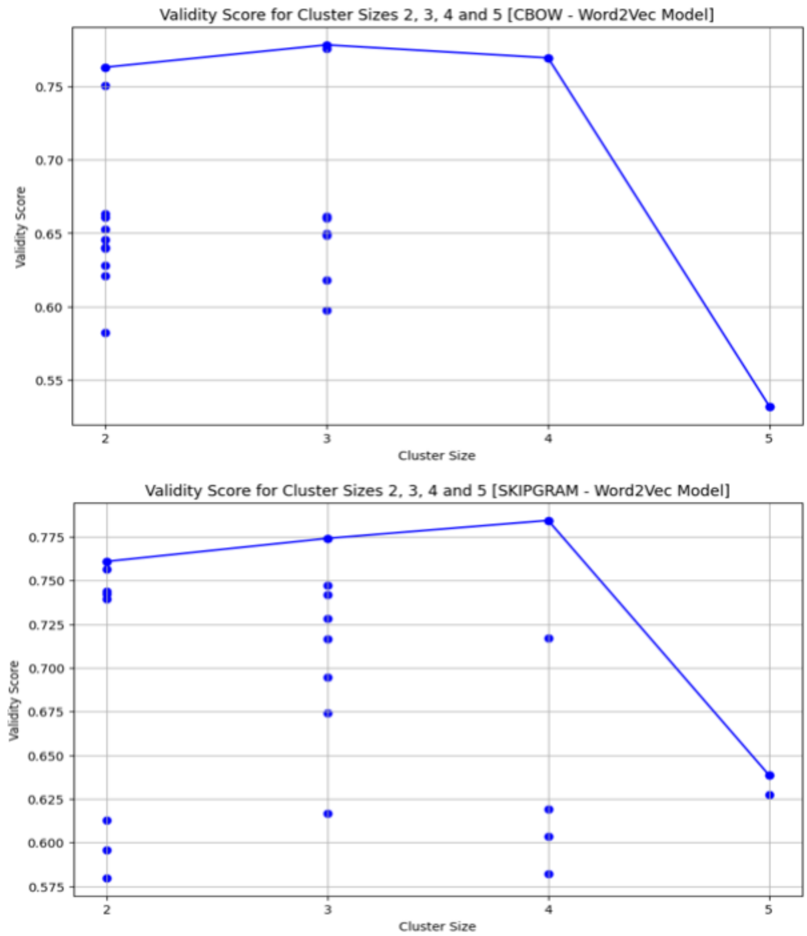


Figure 4.4: Validity Scores for different Cluster Sizes produced after HDBSCAN clustering

For the Word2Vec model, the highest validity score was attained when HDBSCAN clustering produced four clusters. In contrast, the Skip-Gram model achieved its peak validity score with five clusters. It's important to note that these validity scores are the maximum values obtained from a range of hyperparameter combinations, as detailed in *Table 4.1.* For each resulting cluster size from specific set of hyperparameter combination, the maximum validity score are selected to represent the best-quality clustering to determine the optimal number of clusters for each Word2Vec embedding technique. Finally, after filtering the results from every set of hyperparameter combination (Word2Vec + HDBSCAN) based on maximum validity score achieved for each Word2Vec embedding technique, below are values of corresponding hyperparameters:

Table 4.2: Optimal Hyperparameter Values for SkipGram and CBOW Models + HDBSCAN

Hyperparameter	Best SKIPGRAM + HDBSCAN Value	Best CBOW + HDBSCAN Value
Vector Size	100	100
Window	5	2
Min Count	10	10
Min Cluster Size	2	3
Min Samples	8	1
Cluster Selection Epsilon	0.2	0.2
Validity Score	0.7843	0.7780
Number of Clusters	5	4

4.2.2 Hyperparameter Tuning Result for Sentence BERT and HDBSCAN

Same range of hyperparameter values are used for HDBSCAN as mentioned in *Table 4.1* along with pre-trained Sentence BERT encoding model `'all-mpnet-base-v2'` [79]. After optimizing HDBSCAN clustering algorithm using grid-search hyperparameter tuning, the best set of hyperparameter values and number of optimal clusters found with highest validity score of HDBSCAN is as follows:

Table 4.3: Optimized HDBSCAN Hyperparameters with Sentence BERT

Hyperparameter	Optimized Value
Min Cluster Size	1
Cluster Selection Epsilon	1.5
Min Samples	4
Validity Score	0.7093
Number of Clusters	4

4.2.3 Hyperparameter Tuning Result for Doc2Vec and HDBSCAN

Below given are range of Doc2Vec Hyperparameter values used in conjunction with HDBSCAN:

Table 4.4: Range of Hyperparameters for Doc2Vec

Doc2Vec Parameters	Range
Vector Size	50, 100, 200
Window Size	2, 5, 10
Minimum Count	2, 5, 10

The bar graph below presents the average validity scores of clusters generated through various combinations of Doc2Vec and HDBSCAN hyperparameters, arranged in descending order of average cluster quality after clustering. It indicates that the most optimal quality clustering is achieved when the following settings are used: Vector Size = 50, Window Size = 2, and Minimum Count = 5 for Doc2Vec in conjunction with HDBSCAN.

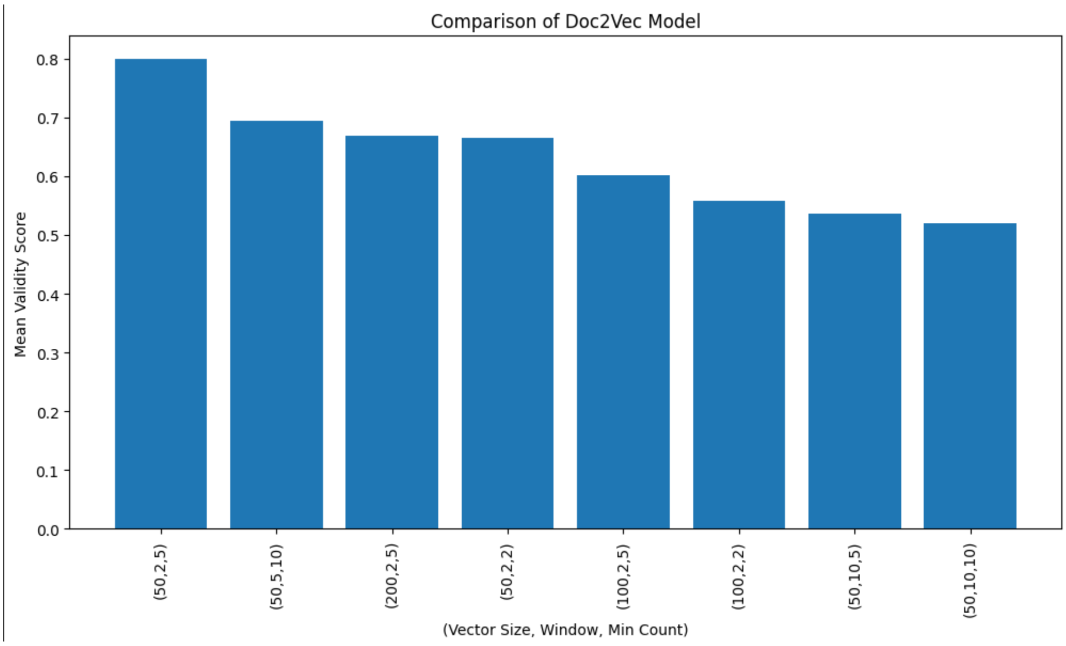


Figure 4.5: Performance comparison for Doc2Vec Hyperparameters after HDBSCAN clustering

After grid-search optimization of Doc2Vec + HDBSCAN, the optimized values for set of hyperparameters are found to be as given below:

Table 4.5: Optimal Hyperparameter Values for Doc2Vec + HDBSCAN

Hyperparameter	**Best Doc2Vec + HDBSCAN Value**
Vector Size	50.0
Window	2.0
Min Count	5.0
Min Cluster Size	2.0
Min Samples	13.0
Cluster Selection Epsilon	0.0
Validity Score	0.7989
Number of Clusters	4.0

4.2.4 Final Selection of Encoding Technique

In the preceding sections (*4.2.1, 4.2.2, and 4.2.3*), three encoding techniques in conjunction with HDBSCAN clustering were evaluated and compared to determine the optimal number of job description clusters. Once the optimal number of high-quality clusters and the respective optimal values for each encoding technique were determined (as given in previously *Table 4.2, 4.3, 4.5*), **final selection of the encoding technique is done using two key clustering evaluation metrics: the Silhouette Score (which should be maximum) and the Davies-Bouldin Index (which should be minimum).**

Table 4.6: Comparison of Clustering Metrics for Different Encoding Techniques

Encoding Technique	**Silhouette Score**	**Davies-Bouldin Index**	**Optimal Clusters**
SkipGram Word2Vec	0.483	0.721	5
CBOW Word2Vec	0.432	0.612	4
Sentence BERT	0.483	0.389	4
Doc2Vec	0.483	0.461	4

From the comparative analysis presented in above *Table 4.6*, it is evident that **Sentence BERT outperforms the other encoding techniques in both Silhouette Score and the Davies-Bouldin Index**, as highlighted in green.

Ultimately, after identifying the optimal number of high-quality clusters for job descriptions using Sentence-BERT and HDBSCAN, these descriptions were re-clustered using the K-Means Algorithm. This step has been done to facilitate future assignment of resumes, encoded with the same Sentence-BERT model, to appropriate cluster labels after training Sentence-BERT encoded job descriptions using the K-Means Algorithm.

After clustering Sentence-BERT encoded job descriptions using K-Means Clustering Algorithm, **the Silhouette Score achieved is 0.581**, indicating good separation between encoded job descriptions as data points closer to other data points in their respective clusters and farther away from data points in other clusters.

Given figure shows clusters of Sentence-BERT encoded Job description after applying K-Means clustering algorithm, visualized in 2-Dimension using UMAP:

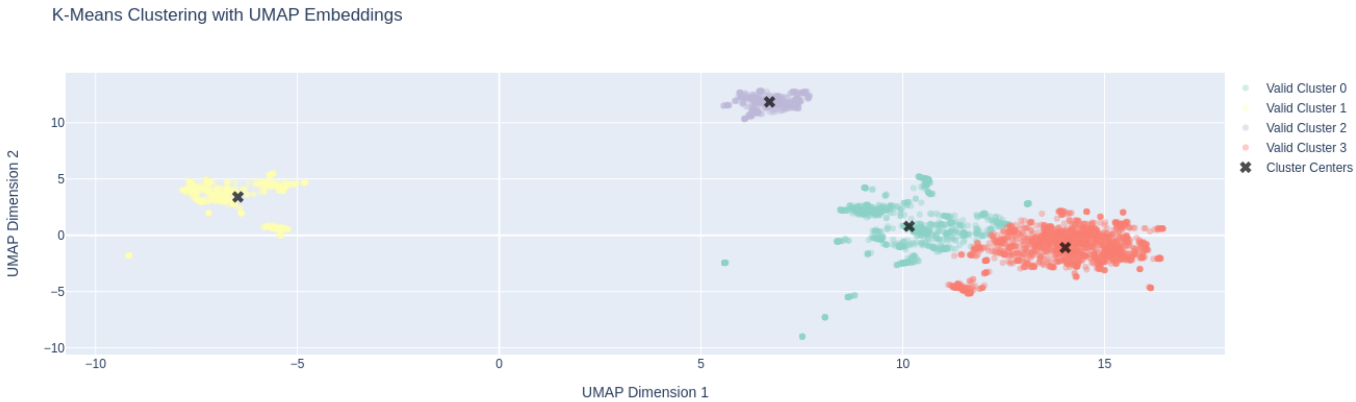


Figure 4.6: K-Means clusters of Sentence-BERT Encoded Job Descriptions

After manual inspection, each job description category was assigned following cluster label using trained K-Mean algorithm:

Table 4.7: Cluster Labels Assigned to Each Job Description Category

Job Description Category	**Assigned Cluster Label**
Product Manager	2
Software Developer	3 (some as 0)
Machine Learning	0 (some as 3)
Chartered Accountant	1

In the above clustering visualization, job descriptions related to **Software Development are depicted in red**, and those associated with **Machine Learning are shown in green**. Given the inherent similarities between these two fields, their clusters are situated close to each other. In contrast, clusters representing **Product Management (shown in purple)** and **Chartered Accountancy (displayed in yellow)** are noticeably distinct and distant from each other, signifying a lack of shared characteristics between these roles. Also, within each cluster, job descriptions related to a specific category are densely grouped together. This visual evidence confirms the effectiveness of using Sentence-BERT for encoding job descriptions and using K-Means clustering to assign appropriate labels to resume that may belong to one of a specific job category

4.3 Document Similarity Experiments and Results

As outlined in Section 3.4.2, the efficacy of context capture in encoding or embedding vectors for the task of document similarity is assessed through the use of a Support Vector Machine classification algorithm. The metric selected for this evaluation is accuracy, applicable to each encoding technique— **Doc2Vec, Sentence-BERT, and BERT + TF-IDF**. **The underlying rationale is that effective context capture by an encoding should result in accurate predictions of the job category type**. Prior to experimentation, only Doc2Vec encoding technique is optimized via grid-search hyperparameter tuning, utilizing a range of hyperparameter values as provided in the *Table 4.8*.

Table 4.8: Range of Hyperparameters for Doc2Vec

Hyperparameter	**Range of Values**
Vector Size	50, 100, 150, 200, 300, 384
Epochs	10, 20, 30, 50, 100, 500
Min Count	2, 5, 10, 15
Window	2, 5, 10, 15

A summary of the accuracy in predicting job categories for respective encoding technique after capturing job description context with encoding vectors, is given below.

Table 4.9: Summary of Validation and Test Accuracy for Different Encoding Techniques

Encoding Technique	Validation Accuracy	Test Accuracy	Best Parameters
Doc2vec	69.0%	69.4%	'**vector_size**': 384, '**min_count**': 5, '**epochs**': 100, '**window**': 2
Sentence Transformer	98.0%	98.0%	–
BERT + TF-IDF	95.0%	94.0%	–

The evaluation summary above indicates that **Sentence-BERT is the most effective at capturing the context from job descriptions such that it could predict job categories with accuracy rate of 98% after encoding with Sentence-BERT**. It means that later resume could be encoded with same Sentence-BERT encoding technique, so that similarity with Sentence-BERT Job Description can be calculated using cosine similarity as done in *Section 3.6.2*

4.4 Skill Extraction Experiments and Result

As described in *Section 3.5.1*, LSTM model is used for classifying the extracted phrases as 'skill' or 'no-skill' for determining if the extracted phrase from job description is a skill or not. For this, LSTM model is optimized with range of its associated hyperparameter values as given in the table below:

Table 4.10: Range of Hyperparameter Values for LSTM Optimization

Hyperparameter	Range
embedding_dims	Integer: [50, 300]
Spatial_Dropout	Float: [0.1, 0.3], step=0.05
n_layers	Integer: [1, 5]
lstm_{i}_units	Integer: [8, 100], step=32
Dropout_rate_{i}	Float: [0, 0.5], step=0.1
layer_2_neurons	Integer: [8, 100], step=32
Dropout_rate_last	Float: [0, 0.5], step=0.1
learning_rate	Choice: [1e-2, 1e-3, 1e-4]

After Bayesian optimization [4], optimized LSTM architecture is as below:

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 16, 190)	2376710
spatial_dropout1d (SpatialDropout1D)	(None, 16, 190)	0
lstm (LSTM)	(None, 16, 40)	36960
dropout (Dropout)	(None, 16, 40)	0
lstm_1 (LSTM)	(None, 16, 8)	1568
dropout_1 (Dropout)	(None, 16, 8)	0
lstm_2 (LSTM)	(None, 16, 8)	544
dropout_2 (Dropout)	(None, 16, 8)	0
lstm_3 (LSTM)	(None, 16, 8)	544
dropout_3 (Dropout)	(None, 16, 8)	0
lstm_4 (LSTM)	(None, 16, 8)	544
dropout_4 (Dropout)	(None, 16, 8)	0
lstm_5 (LSTM)	(None, 72)	23328
dropout_5 (Dropout)	(None, 72)	0
dense (Dense)	(None, 1)	73

Total params: 2,440,271
Trainable params: 2,440,271
Non-trainable params: 0

Figure 4.7: Optimized LSTM architecture for classifying phrases as 'skill' / 'no-skill' and extracting skills from Job Descriptions

This optimized LSTM architecture achieved the Test Accuracy of 88.51% in classifying the extracted phrase from job descriptions into 'skill' or 'no-skill'. Based on the predictions of extracted phrase as 'skills', all the skill-set could be extracted from the job descriptions.

4.5 Resume to Job Descriptions Matching Result

Initially, to evaluate the effectiveness of pre-trained K-Means clustering algorithm, trained on different categories of job descriptions (as described in *Section 4.2.4*), in assigning the learned cluster labels to resume which is contextually similar to job descriptions in that cluster, the cluster assignment is done on all resume in the database belonging to different categories, as mentioned in *Section 4.1.* The result of this cluster label assignment to all resume in the database is given below:

[SEE APPENDIX FOR RESUME MATCHING TO JOB DESCRIPTION PIPELINE CODE]

	_id	index	category	text	parsed_resume	predicted_cluster_label
0	649d8da410170921a743bf10	SD_resume_5.pdf	software developer	rsum david baumgold david baumgold fullstack w...	{'name': 'David Baumgold', 'email': 'david@dav...	0
1	649d8da610170921a743bf11	SD_resume_8.pdf	software developer	rakesh neela resume r k e h n e e l h sanfanci...	{'name': 'STATE UNIVERSITY', 'email': 'rakeshn...	0
2	649d8da810170921a743bf12	SD_resume_2.pdf	software developer	cobaohieuresume co bao hieu hochiminh city vn ...	{'name': 'Bao Hieu', 'email': 'cobaohieu@gmail...	0
3	649d8da910170921a743bf13	SD_resume_6.pdf	software developer	andrew dillon resume andrew dillon 402 6317966...	{'name': 'cid:57)ORK E(cid:58)PER(cid:882)ENCE...	0
4	649d8daa10170921a743bf14	SD_resume_4.pdf	software developer	joel verhagen seattle washington joelverhagen ...	{'name': 'Joel Verhagen', 'email': 'joel.verha...	3
5	649d8dab10170921a743bf15	SD_resume_9.pdf	software developer	shubham singh junior software developers shubh...	{'name': 'S SHUBHAM', 'email': 'shubh2014shiv@...	0
6	649d8dad10170921a743bf16	SD_resume_1.pdf	software developer	resume ayush gupta ayushg3112 919013363330 ski...	{'name': 'Ayush Gupta', 'email': 'AyushG3112@g...	0
7	649d8dae10170921a743bf17	SD_resume_3.pdf	software developer	resume ayush gupta ayushg3112 919013363330 ski...	{'name': 'Ayush Gupta', 'email': 'AyushG3112@g...	0
8	649d8db010170921a743bf18	SD_resume_7.pdf	software developer	dipta das software engineer dipta670 1 2547179...	{'name': 'DIPTA DAS', 'email': 'dipta670@gmail...	0
9	649d8db110170921a743bf19	PM_resume_1.pdf	product manager	kiran kumar parasa 1234 apple street pune maha...	{'name': 'Kiran Kumar', 'email': 'YourName@gma...	2
10	649d8db210170921a743bf1a	ML_resume_5.pdf	machine learning	meschiaricv data science leader work crossfunc...	{'name': 'Stefano Meschiari', 'email': 'stefan...	0
11	649d8db410170921a743bf1b	ML_resume_4.pdf	machine learning	data sciencemachine learning resume name phone...	{'name': 'Email Waltham', 'email': None, 'mobi...	0
12	649d8db510170921a743bf1c	ML_resume_2.pdf	machine learning	1 tung thanh le website us permanent residency...	{'name': 'Tung Thanh', 'email': 'ttungl@gmail....	0
13	649d8db610170921a743bf1d	ML_resume_3.pdf	machine learning	abdallah bashir 447493734669 london uk educat...	{'name': 'Abdallah Bashir', 'email': 'TEDxYout...	0
14	649d8db810170921a743bf1e	ML_resume_6.pdf	machine learning	resume juan jose carin 1 2 juan jose carin dat...	{'name': 'Juan Jose', 'email': 'juanjose.carin...	0
15	649d8db910170921a743bf1f	ML_resume_1.pdf	machine learning	yunlong jiao machine learning scientist london...	{'name': 'Yunlong Jiao', 'email': 'yljiao.ustc...	0
16	649d8dbb10170921a743bf20	CA_resume_1.pdf	chartered accountant	fugeecv simon mukuze cv page 1 5 curriculum vi...	{'name': 'CURRICULUM VITAE', 'email': 'smukuze...	1
17	649d8dbc10170921a743bf21	CA_resume_5.pdf	chartered accountant	ethan miller 456 elm st london uk 123 4567891 ...	{'name': 'Ethan Miller', 'email': 'ethanmiller...	1
18	649d8dbd10170921a743bf22	CA_resume_3.pdf	chartered accountant	john doe 123 main st middlesbrough uk 123 4567...	{'name': 'John Doe', 'email': 'johndoe@email.c...	1
19	649d8dbe10170921a743bf23	CA_resume_2.pdf	chartered accountant	microsoft word beautifulresumeforcasample2doc ...	{'name': 'CA Resume', 'email': 'nehakumar@abc....	0
20	649d8dbf10170921a743bf24	CA_resume_4.pdf	chartered accountant	emily johnson 123 beech rd manchester uk 123 4...	{'name': 'Emily Johnson', 'email': 'emilyjohns...	1

Figure 4.8: Cluster assignment to all the resumes available in the database using trained K-Means clustering Algorithm

As shown in the above figure 4.8, upon manual inspection, **the labels assigned to resume belonging to respective categories matches with cluster labels assigned earlier (*Table 4.7*) to the job descriptions belonging to their corresponding categories**. This confirms the effectiveness of combination of Sentence-BERT and the K-means clustering algorithm to effectively cluster the job descriptions and contextually assign the corresponding labels to resume belonging to the categories on which these algorithms are trained on same categories of job descriptions in unsupervised approach.

Furthermore, to assess the efficacy of the pipeline, mentioned in Methodology chapter, in matching a job seeker's resume from all available job descriptions, a single resume was deliberately chosen manually for the evaluation. This particular publicly available resume, [RESUME LINK] , is rich in both skills and past work experience, providing a comprehensive test case for the proposed matching algorithm.

After subjecting the given resume to the same preprocessing steps for clustering, it was encoded using the same Sentence-BERT algorithm to produce embedding vectors of the resume. These vectors were then reduced to two dimensions for compatibility with trained K-means algorithm as described in Section 3.4 on Document Clustering. **The resume was subsequently assigned the cluster label as '0', indicating that it is most closely contextually related to the Machine Learning category of the job descriptions.**

Next, using the stage II pipeline, as described earlier in section 3.4, both the given resume and all the job descriptions specifically in cluster '0' were subjected to same preprocessing steps for document similarity, noise removal and Sentence-BERT encoding to convert them into clean and consistent embedding vectors. Subsequently, matching scores measuring the overall context similarity between given resume and all job descriptions in cluster '0' were assessed using the cosine similarity. These matching score for resume again each job description IDs were then arranged in descending order as shown below:

Job Description ID	Document Level Matching Score between Given Resume and Respective Job Description
649cd599eed36cf2eea931d0	0.8455585
649cd599eed36cf2eea9323b	0.8117122
649cd599eed36cf2eea931e3	0.8101116
649cd599eed36cf2eea931ee	0.7979156
649cd599eed36cf2eea93201	0.79223037
649cd599eed36cf2eea931c3	0.7915017
649cd599eed36cf2eea9321d	0.7873909
649cd599eed36cf2eea931b9	0.7708134
649cd599eed36cf2eea93205	0.76861435
649cd599eed36cf2eea931c9	0.76466155
649cd599eed36cf2eea931ce	0.76216847
649cd599eed36cf2eea931bf	0.7613545
649cd599eed36cf2eea93229	0.7548926
649cd599eed36cf2eea931df	0.7539602
649cd599eed36cf2eea931ed	0.75144166
649cd599eed36cf2eea931c6	0.7512038
649cd599eed36cf2eea9320f	0.74924433
649cd599eed36cf2eea931b7	0.74920785
649cd599eed36cf2eea93238	0.74920785
649cd599eed36cf2eea93209	0.74913865
649cd599eed36cf2eea931dd	0.74830854
649cd599eed36cf2eea931d5	0.74495304
649cd599eed36cf2eea931b2	0.7442357
649cd599eed36cf2eea9323a	0.74357253
649cd599eed36cf2eea93220	0.74176717
... (SO ON)	
649cd599eed36cf2eea92e19	0.3127549
649cd599eed36cf2eea92fe4	0.3107193
649cd599eed36cf2eea92de6	0.3084779
649cd598eed36cf2eea929f8	0.30790502
649cd598eed36cf2eea92b13	0.29340327

Figure 4.9: Document Level Matching Score Results arranged in descending order against each Job description ID

Next, in stage III, skills from the job descriptions were extracted using same LSTM based skill extraction pipeline as described in *Section 3.5* and skills from resume were extracted using ResumeParser python library, as mentioned earlier in *Section 3.6.3.* The skills from both job descriptions and given resume was encoded using BERT Transformer, followed by determining the semantic similarity between list of extracted skill from both. Given below are the matching scores arranged in descending order of skill-based similarity from each job description ID for the given resume.

Job Description ID	Document Level Matching Score between Given Resume and Respective Job Description
649cd599eed36cf2eea931d0	0.8455585
649cd599eed36cf2eea9323b	0.8117122
649cd599eed36cf2eea931e3	0.8101116
649cd599eed36cf2eea931ee	0.7979156
649cd599eed36cf2eea93201	0.79223037
649cd599eed36cf2eea931c3	0.7915017
649cd599eed36cf2eea9321d	0.7873909
649cd599eed36cf2eea931b9	0.7708134
649cd599eed36cf2eea93205	0.76861435
649cd599eed36cf2eea931c9	0.76466155
649cd599eed36cf2eea931ce	0.76216847
649cd599eed36cf2eea931bf	0.7613545
649cd599eed36cf2eea93229	0.7548926
649cd599eed36cf2eea931df	0.7539602
649cd599eed36cf2eea931ed	0.75144166
649cd599eed36cf2eea931c6	0.7512038
649cd599eed36cf2eea9320f	0.74924433
649cd599eed36cf2eea931b7	0.74920785
649cd599eed36cf2eea93238	0.74920785
649cd599eed36cf2eea93209	0.74913865
649cd599eed36cf2eea931dd	0.74830854
649cd599eed36cf2eea931d5	0.74495304
649cd599eed36cf2eea931b2	0.7442357
649cd599eed36cf2eea9323a	0.74357253
649cd599eed36cf2eea93220	0.74176717
... (SO ON)	
649cd599eed36cf2eea92e19	0.3127549
649cd599eed36cf2eea92fe4	0.3107193
649cd599eed36cf2eea92de6	0.3084779
649cd598eed36cf2eea929f8	0.30790502
649cd598eed36cf2eea92b13	0.29340327

Figure 4.10: Skill-set Level Matching Score Results arranged in descending order against each Job description ID

The matching scores from previous stage II and stage III were averaged against each corresponding Job Description IDs as shown below:

	Job Description ID	Matching Score based on Resume to job Descriptions Similarity_document	Matching Score based on Resume to job Descriptions Similarity_skills	average_matching_score
0	649cd599eed36cf2eea931d0	0.845559	0.611372	0.728465
1	649cd599eed36cf2eea9323b	0.811712	0.635294	0.723503
2	649cd599eed36cf2eea931e3	0.810112	0.625837	0.717974
20	649cd599eed36cf2eea931dd	0.748309	0.679555	0.713932
6	649cd599eed36cf2eea9321d	0.787391	0.638697	0.713044
3	649cd599eed36cf2eea931ee	0.797916	0.621333	0.709624
4	649cd599eed36cf2eea93201	0.792230	0.623492	0.707861
9	649cd599eed36cf2eea931c9	0.764662	0.648402	0.706532
15	649cd599eed36cf2eea931c6	0.751204	0.656969	0.704086
5	649cd599eed36cf2eea931c3	0.791502	0.609848	0.700675

Figure 4.11: Average Matching Scores for each Job Description ID

Based on top 10 average matching scores for each job description IDs from above, the job descriptions are recommended to the job seeker.

Finally, to assess the alignment between the provided resume and the recommended job descriptions retrieved by whole pipeline presented in the project, word clouds [103] for both are generated. The word cloud is a visual representation that displays the most frequently occurring terms in a text corpus, making it easier to spot common themes or patterns. By comparing the word cloud for the cleaned resume with that of the top 10 recommended job descriptions, the extent to which they are closely related in terms of skills, experience, and other relevant keywords could be visually gauged.

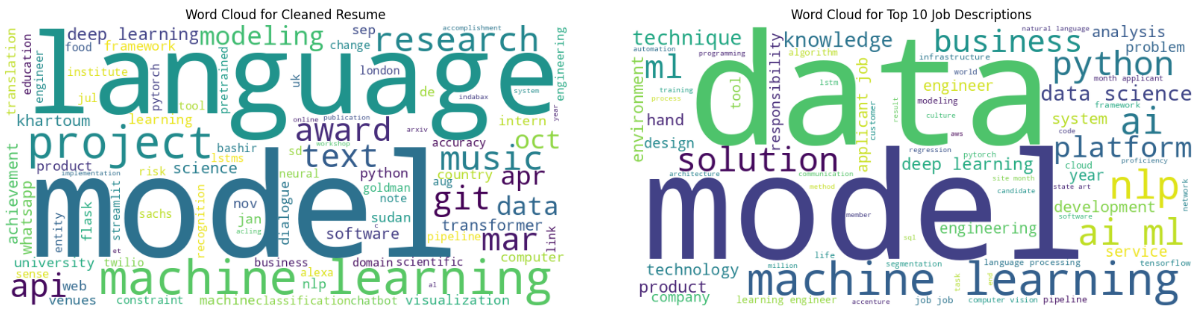


Figure 4.12: Word Clouds for both given Resume (left) and Top 10 Recommended Job Descriptions (right)

Since the most prominent terms like *"model"*, *"language"*, *"NLP"*, *"text"*, *"python"*, *"machine learning"*, *"deep learning"*, *"data"* etc in both word clouds overlap significantly semantically, it is a satisfactory indicator that three-stage context-aware matching pipeline is effectively recommending job descriptions that are well-aligned with the candidate's resume.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Finding a job in today's competitive job market is difficult, but finding the proper one is much more difficult. The variety in skill requirements and expectations has increased more than ever as a result of the growth of job postings on websites like Indeed and LinkedIn. Job seekers searching for the job frequently struggle to decide which job posting best fits their skills and goals. In effort towards solving this problem, this thesis introduced and explored a three-stage context based approach to match the job seeker's resume from available pool of online job advertisements.

While keyword-based approaches have long been widely used for quickly matching resumes with job descriptions, their limitations are becoming more apparent. These traditional techniques often neglect the nuanced aspects of a job seeker's experience and qualifications, fixating only on the presence of specific "keywords" resulting in overlooking the nuances of a job seeker's experience and skills. To address these shortcomings, the current research experimented with a more nuanced, three-stage approach that meets the complexities of today's job market by taking context-based approach in matching process. To facilitate this, the research initially clustered job descriptions based on contextual similarity in stage I. Subsequently, resume predicted to belong to a specific cluster contextually, matching with job descriptions was further refined through whole contextual and skill-based semantic matching within the given cluster in stage II and III. The average matching score was derived from the second and third stages, and the top 10 job descriptions with the highest contextual matching scores were recommended to the job seeker. **Through experimentation, it was found that Sentence-BERT was**

most effective in capturing the essential context of job descriptions during the clustering and semantic similarity matching phases (Stages I and II).

Additionally, the research offered more than just theoretical contributions as the proposed and experimented pipeline has practical business applications, particularly when deployed on cloud computing platforms like AWS, enabling large-scale resume-to-job-description matching.

5.2 Future Works

While the primary focus of this research has been on developing a context-aware pipeline for matching resumes to job descriptions, there are several considerations for future work that could extend and enrich the scope of this project:

- **Model Generalization Across Diverse Job Categories:** Due to time and resource constraints, this study concentrated on just four job categories: Product Management, Chartered Accountancy, Machine Learning, and Software Development. Future work could broaden the scope to include a greater variety of job categories, thereby making the job recommendation system more versatile and applicable to a wider audience of job seekers.

- **Multi-Language Support:** The current implementation utilizes transformer models to understand and capture the context of job descriptions and resumes in English. Given the global reach of online job portals, future work could be adapted to support multiple languages, making the service more accessible to job seekers worldwide.

- **Fairness and Bias:** The present approach could be subjected to further research to assess its fairness and potential biases, particularly when it comes to recommending job descriptions to job seekers from minority groups. Identifying and mitigating such biases would make the system more equitable.

- **Legal and Ethical Considerations:** Given that personal or business information is inherently part of any resume or job description, future work should consider implementing mechanisms to address the legal and ethical challenges that come with automated job matching. This includes concerns related to data privacy and potential discrimination.

- **Cloud Deployment and Scalability:** The three-stage context-aware matching pipeline developed in this thesis has the potential for scalability. Future work could involve deploying this system on a cloud-based infrastructure like AWS, allowing it to serve a larger number of job seekers efficiently.

- **Skill Gap Analysis:** Another prospective extension of this model involves identifying and quantifying skill gaps for job seekers. This feature would not only match candidates to suitable roles but also provide valuable insights for those looking to enhance specific skills, thereby increasing their marketability for particular positions.

- **Evaluation Metrics for Job Description Retrieval:** The current methodology primarily uses word clouds to visualize and validate the effectiveness of the job description retrieval and recommendation system for given resume. While this approach provides a qualitative assessment, it is somewhat limited in rigorously quantifying the accuracy, relevance, and reliability of the retrieved job descriptions. Therefore, future research could focus on implementing and testing various evaluation metrics to provide a more robust, quantitative analysis.

References

[1] Indeed, *What are keywords in job applications? — indeed.com*, 2023. [Online]. Available: https://www.indeed.com/career-advice/finding-a-job/what-are-keywords-in-job-applications.

[2] J. Malinowski, T. Keim, O. Wendt, and T. Weitzel, "Matching people and jobs: A bilateral recommendation approach," in *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, vol. 6, 2006, pp. 137c–137c. DOI: 10.1109/HICSS.2006.266.

[3] A. Singh, R. Catherine, K. V. Ramanan, V. Chenthamarakshan, and N. Kambhatla, "Prospect: A system for screening candidates for recruitment," *Proceedings of the 19th ACM international conference on Information and knowledge management*, 2010. [Online]. Available: https://api.semanticscholar.org/CorpusID:5276445.

[4] J. Snoek, H. Larochelle, and R. P. Adams, *Practical bayesian optimization of machine learning algorithms*, 2012. arXiv: 1206.2944 [stat.ML].

[5] H. S. Writer, *Candidate Match Score: A Complete Guide — techrseries.com*, https://techrseries.com/recruitment-and-on-boarding/candidate-match-score-a-complete-guide/, [Accessed 17-09-2023], 2023.

[6] IBM, *What is natural language processing? — ibm*, (Accessed on 09/13/2023). [Online]. Available: https://www.ibm.com/topics/natural-language-processing.

[7] MongoDB, *What is unstructured data? — mongodb*, (Accessed on 09/13/2023). [Online]. Available: https://www.mongodb.com/unstructured-data.

[8] StackRoute, *(10) the evolution of natural language processing: From rule-based systems to transformers — linkedin*, (Accessed on 09/13/2023), 2023. [Online]. Available: https://www.linkedin.com/pulse/evolution-natural-language-processing-from-rule-based-systems/.

[9] I. Sutskever, O. Vinyals, and Q. V. Le, *Sequence to sequence learning with neural networks*, 2014. DOI: 10.48550/ARXIV.1409.3215. [Online]. Available: https://arxiv.org/abs/1409.3215.

[10] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. arXiv: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762.

[11] R. M. Schmidt, "Recurrent neural networks (rnns): A gentle introduction and overview," *CoRR*, vol. abs/1912.05911, 2019. arXiv: 1912.05911. [Online]. Available: http://arxiv.org/abs/1912.05911.

[12] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *CoRR*, vol. abs/1808.03314, 2018. arXiv: 1808.03314. [Online]. Available: http://arxiv.org/abs/1808.03314.

[13] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. arXiv: 1412.3555. [Online]. Available: http://arxiv.org/abs/1412.3555.

[14] D. S. Batista, *The attention mechanism in natural language processing*, https://www.davidsbatista.net/blog/2020/01/25/Attention-seq2seq/, (Accessed on 09/13/2023), 2020.

[15] A. Majeed, *Deep learning - how seq2seq context vector is generated? - stack overflow*, https://stackoverflow.com/questions/75545619/how-seq2seq-context-vector-is-generated, (Accessed on 09/13/2023), 2023.

[16] A. Zbiciak and T. Markiewicz, "A new extraordinary means of appeal in the polish criminal procedure: The basic principles of a fair trial and a complaint against a cassatory judgment," en, *Access to Justice in Eastern Europe*, vol. 6, no. 2, pp. 1–18, Mar. 2023.

[17] K. M.Tarwani, and S. Edem, "Survey on recurrent neural network in natural language processing," *International Journal of Engineering Trends and Technology*, vol. 48, no. 6, pp. 301–304, Jun. 2017. DOI: 10.14445/22315381/ijett-v48p253. [Online]. Available: https://doi.org/10.14445/22315381/ijett-v48p253.

[18] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132 306, Mar. 2020. DOI: 10.1016/j.physd.2019.132306. [Online]. Available: https://doi.org/10.1016/j.physd.2019.132306.

[19] A. Gillioz, J. Casas, E. Mugellini, and O. A. Khaled, "Overview of the transformer-based models for NLP tasks," in *Proceedings of the 2020 Federated Conference on Computer Science and Information Systems*, IEEE, Sep. 2020. DOI: 10.15439/2020f20. [Online]. Available: https://doi.org/10.15439/2020f20.

[20] Pinecone, *Sentence transformers: Meanings in disguise — pinecone*, https://www.pinecone.io/learn/series/nlp/sentence-embeddings/, (Accessed on 09/13/2023).

[21] A. Faure, *An introduction to bert*, https://www.amauryfaure.com/posts/introduction-bert, (Accessed on 09/13/2023), 2021.

[22] J. Uszkoreit, *Transformer: A novel neural network architecture for language understanding – google research blog*, https://blog.research.google/2017/08/transformer-novel-neural-network.html, (Accessed on 09/13/2023), 2017.

[23] J. Alammar, *The illustrated transformer – jay alammar – visualizing machine learning one concept at a time.* https://jalammar.github.io/illustrated-transformer/, (Accessed on 09/13/2023), 2018.

[24] Jansen, *Why people always say the transformer is parallelizable while the self-attention layer still depends on outputs of all time steps to calculate? - artificial intelligence stack exchange*, https://ai.stackexchange.com/questions/29903/why-people-always-say-the-transformer-is-parallelizable-while-the-self-attention, (Accessed on 09/13/2023), 2021.

[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2018. DOI: 10.48550/ARXIV.1810.04805. [Online]. Available: https://arxiv.org/abs/1810.04805.

[26] S. González-Carvajal and E. C. Garrido-Merchán, "Comparing bert against traditional machine learning text classification," 2020. DOI: 10.48550/ARXIV.2005.13012. [Online]. Available: https://arxiv.org/abs/2005.13012.

[27] I. Hamdine, *Fine-tuning bert for semantic textual similarity with transformers in python - python code*, https://thepythoncode.com/article/finetune-bert-for-semantic-textual-similarity-in-python, (Accessed on 09/13/2023), 2023.

[28] J. A. Alzubi, R. Jain, A. Singh, P. Parwekar, and M. Gupta, "COBERT: COVID-19 question answering system using BERT," *Arabian Journal for Science and Engineering*, vol. 48, no. 8, pp. 11 003–11 013, Jun. 2021. DOI: 10.1007/s13369-021-05810-5. [Online]. Available: https://doi.org/10.1007/s13369-021-05810-5.

[29] H. Mohebbi, A. Modarressi, and M. T. Pilehvar, *Exploring the role of bert token representations to explain sentence probing results*, 2021. DOI: 10.48550/ARXIV.2104.01477. [Online]. Available: https://arxiv.org/abs/2104.01477.

[30] HuggingFace, *What is sentence similarity? - hugging face*, https://huggingface.co/tasks/sentence-similarity, (Accessed on 09/13/2023).

[31] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Conference on Empirical Methods in Natural Language Processing*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:201646309.

[32] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," in *Proceedings of the 6th International Conference on Neural Information Processing Systems*, ser. NIPS'93, Denver, Colorado: Morgan Kaufmann Publishers Inc., 1993, 737–744.

[33] W. A. Qader, M. M. Ameen, and B. I. Ahmed, "An overview of bag of words;importance, implementation, applications, and challenges," in *2019*

International Engineering Conference (IEC), 2019, pp. 200–204. DOI: 10.1109/IEC47844.2019.8950616.

[34] *Tf–idf - Wikipedia — en.wikipedia.org*, https://en.wikipedia.org/wiki/Tf-idf, [Accessed 17-09-2023].

[35] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013. DOI: 10.48550/ARXIV.1301.3781. [Online]. Available: https://arxiv.org/abs/1301.3781.

[36] P. Lison and A. Kutuzov, "Redefining context windows for word embedding models: An experimental study," *CoRR*, vol. abs/1704.05781, 2017. arXiv: 1704.05781. [Online]. Available: http://arxiv.org/abs/1704.05781.

[37] S. A. Alsaif, M. S. Hidri, H. A. Eleraky, I. Ferjani, and R. Amami, "Learning-based matched representation system for job recommendation," *Computers*, vol. 11, no. 11, p. 161, Nov. 2022. DOI: 10.3390/computers11110161. [Online]. Available: https://doi.org/10.3390/computers11110161.

[38] M. Lukauskas, V. Šarkauskaitė, V. Pilinkienė, A. Stundžienė, A. Grybauskas, and J. Bruneckienė, "Enhancing skills demand understanding through job ad segmentation using NLP and clustering techniques," *Applied Sciences*, vol. 13, no. 10, p. 6119, May 2023. DOI: 10.3390/app13106119. [Online]. Available: https://doi.org/10.3390/app13106119.

[39] C. Syms, "Principal components analysis," in *Encyclopedia of Ecology*, S. E. Jørgensen and B. D. Fath, Eds., Oxford: Academic Press, 2008, pp. 2940–2949, ISBN: 978-0-08-045405-4. DOI: https://doi.org/10.1016/B978-008045405-4.00538-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780080454054005383.

[40] L. McInnes, J. Healy, and J. Melville, *Umap: Uniform manifold approximation and projection for dimension reduction*, 2020. arXiv: 1802.03426 [stat.ML].

[41] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of Machine Learning Research*, vol. 9, no. 11, 2008.

[42] J. Venna and S. Kaski, "Neighborhood preservation in nonlinear projection methods: An experimental study," in *International Conference on Artificial Neural Networks*, 2001. [Online]. Available: https://api.semanticscholar.org/CorpusID:14473245.

[43] S. Na, L. Xumin, and G. Yong, "Research on k-means clustering algorithm: An improved k-means clustering algorithm," in *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, 2010, pp. 63–67. DOI: 10.1109/IITSI.2010.74.

[44] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," ser. KDD'96, Portland, Oregon: AAAI Press, 1996, 226–231.

[45] L. McInnes, J. Healy, and S. Astels, "Hdbscan: Hierarchical density based clustering," *Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017. DOI: 10.21105/joss.00205. [Online]. Available: https://doi.org/10.21105/joss.00205.

[46] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987, ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0377042787901257.

[47] J. C. Rojas Thomas, M. S. Peñas, and M. Mora, "New version of davies-bouldin index for clustering validation based on cylindrical distance," in *2013 32nd International Conference of the Chilean Computer Science Society (SCCC)*, 2013, pp. 49–53. DOI: 10.1109/SCCC.2013.29.

[48] X. Wang and Y. Xu, "An improved index for clustering validation based on silhouette index and calinski-harabasz index," *IOP Conference Series: Materials Science and Engineering*, vol. 569, no. 5, p. 052 024, 2019. DOI: 10.1088/1757-899X/569/5/052024. [Online]. Available: https://dx.doi.org/10.1088/1757-899X/569/5/052024.

[49] T. K, U. V, S. M. Kadiwal, and S. Revanna, "Design and development of machine learning based resume ranking system," *Global Transitions Proceedings*, vol. 3, no. 2, pp. 371–375, Nov. 2022. DOI: 10.1016/j.gltp.2021.10.002. [Online]. Available: https://doi.org/10.1016/j.gltp.2021.10.002.

[50] "A COSINE SIMILARITY-BASED RESUME SCREENING SYSTEM FOR JOB RECRUITMENT," *International Research Journal of Modernization in Engineering Technology and Science*, 2023. DOI: 10.56726/irjmets35945. [Online]. Available: https://doi.org/10.56726%2Firjmets35945.

[51] N. Vanetik and G. Kogan, "Job vacancy ranking with sentence embeddings, keywords, and named entities," *Information*, vol. 14, no. 8, p. 468, 2023. DOI: 10.3390/info14080468. [Online]. Available: https://doi.org/10.3390%2Finfo14080468.

[52] S. Pudasaini, S. Shakya, S. Lamichhane, S. Adhikari, A. Tamang, and S. Adhikari, "Scoring of resume and job description using word2vec and matching them using gale–shapley algorithm," in *Expert Clouds and Applications*, Springer Singapore, 2021, pp. 705–713. DOI: 10.1007/978-981-16-2126-0_55. [Online]. Available: https://doi.org/10.1007%2F978-981-16-2126-0_55.

[53] V. Bhatia, P. Rawat, A. Kumar, and R. R. Shah, "End-to-end resume parsing and finding candidates for a job description using bert," *ArXiv*, vol. abs/1910.03089, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:203905435.

[54] Y. Hou, X. Pan, W. X. Zhao, *et al.*, "Leveraging search history for improving person-job fit," in *Database Systems for Advanced Applications*, Springer International Publishing, 2022, pp. 38–54. DOI: 10.1007/978-3-031-00123-9_3. [Online]. Available: https://doi.org/10.1007/978-3-031-00123-9_3.

[55] S. H. E∗ and M. A. E, "Differential hiring using a combination of NER and word embedding," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 9, no. 1, pp. 1344–1349, 2020. DOI: 10.35940/ijrte.a2400.059120. [Online]. Available: https://doi.org/10.35940/ijrte.a2400.059120.

[56] N. Sharma, *Job skills extraction with lstm and word embeddings*, https://confusedcoders.com/wp-content/uploads/2019/09/Job-Skills-extraction-with-LSTM-and-Word-Embeddings-Nikita-Sharma.pdf, (Accessed on 09/13/2023), 2020.

[57] R. Ajayi, *Remeajayi/ds-job-detective: Text extraction with pos tagging and deep learning(lstms)*, (Accessed on 09/13/2023), 2022. [Online]. Available: https://github.com/RemeAjayi/ds-job-detective.

[58] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, 1735–1780, 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735.

[59] I. Montani, M. Honnibal, M. Honnibal, *et al.*, *Explosion/spacy: V3.6.1: Support for pydantic v2, find-function cli and more*, 2023. DOI: 10.5281/ZENODO.1212303. [Online]. Available: https://zenodo.org/record/1212303.

[60] *Part of speech - wikipedia*, https://en.wikipedia.org/wiki/Part_of_speech, (Accessed on 09/13/2023).

[61] *Semantic similarity - Wikipedia — en.wikipedia.org*, https://en.wikipedia.org/wiki/Semantic_similarity, [Accessed 18-09-2023].

[62] R. Alake, *Understanding cosine similarity and its application — built in*, https://builtin.com/machine-learning/cosine-similarity, (Accessed on 09/14/2023), 2023.

[63] MongoDB, *What is nosql? nosql databases explained — mongodb*, https://www.mongodb.com/nosql-explained, (Accessed on 09/14/2023).

[64] N. Shah and S. Mahajan, "Semantic based document clustering: A detailed review," *International Journal of Computer Applications*, vol. 52, no. 5, pp. 42–52,

Aug. 2012. DOI: 10.5120/8202-1598. [Online]. Available: https://doi.org/10.5120/8202-1598.

[65] N. V. Otten, *The curse of dimensionality and how to overcome it*, https://spotintelligence.com/2022/11/29/curse-of-dimensionality/, (Accessed on 09/14/2023), 2022.

[66] C. Ding, "Dimension reduction techniques for clustering," in *Encyclopedia of Database Systems*, L. LIU and M. T. ÖZSU, Eds. Boston, MA: Springer US, 2009, pp. 846–846, ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_612. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_612.

[67] A. Kulkarni and A. Shivananda, *Natural Language Processing Recipes*. Apress, 2019. DOI: 10.1007/978-1-4842-4267-4. [Online]. Available: https://doi.org/10.1007/978-1-4842-4267-4.

[68] J. Luts, *Using deep learning to extract knowledge from job descriptions - kdnuggets*, https://www.kdnuggets.com/2017/05/deep-learning-extract-knowledge-job-descriptions.html, (Accessed on 09/14/2023).

[69] A. Kaur, P. Kumar, and P. Kumar, "Effect of noise on the performance of clustering techniques," in *2010 International Conference on Networking and Information Technology*, IEEE, Jun. 2010. DOI: 10.1109/icnit.2010.5508461. [Online]. Available: https://doi.org/10.1109/icnit.2010.5508461.

[70] J. Thanaki, *Python Natural Language Processing*. Packt Publishing, 2017, p. 221, ISBN: 9781787285521, 1787285529.

[71] Wikipedia, *Word2vec - wikipedia*, https://en.wikipedia.org/wiki/Word2vec#Context_window, (Accessed on 09/14/2023), 2023.

[72] B. Seegmiller, D. Papanikolaou, and L. Schmidt, *Data and code for "measuring document similarity with weighted averages of word embeddings"*, en, 2022. DOI: 10.3886/E182925V2. [Online]. Available: https://www.openicpsr.org/openicpsr/project/182925/version/V2/view.

[73] T. Adewumi, F. Liwicki, and M. Liwicki, "Word2vec: Optimal hyperparameters and their impact on natural language processing downstream tasks," *Open Computer Science*, vol. 12, no. 1, pp. 134–141, Jan. 2022. DOI: 10.1515/comp-2022-0236. [Online]. Available: https://doi.org/10.1515/comp-2022-0236.

[74] Gensim, *Models.word2vec – word2vec embeddings — gensim*, https://radimrehurek.com/gensim/models/word2vec.html, (Accessed on 09/14/2023), 2022.

[75] H. Caselles-Dupré, F. Lesaint, and J. Royo-Letelier, *Word2vec applied to recommendation: Hyperparameters matter*, 2018. DOI: 10.48550/ARXIV.1804.04212. [Online]. Available: https://arxiv.org/abs/1804.04212.

[76] S. Vajjala, B. Majumder, A. Gupta, and H. Surana, *Practical Natural Language Processing*, First Edition. O'Reilly Media, Inc., 2020, pp. 138–140, ISBN: 978-1-492-05405-4.

[77] S. Li, *Multi-class text classification with doc2vec & logistic regression — by susan li — towards data science*, `https://towardsdatascience.com/multi-class-text-classification-with-doc2vec-logistic-regression-9da9947b43f4`, (Accessed on 09/14/2023), 2018.

[78] Gensim, *Models.doc2vec – doc2vec paragraph embeddings — gensim*, `https://radimrehurek.com/gensim/models/doc2vec.html`, (Accessed on 09/14/2023), 2022.

[79] HuggingFace, *Sentence-transformers/all-mpnet-base-v2 · hugging face*, `https://huggingface.co/sentence-transformers/all-mpnet-base-v2`, (Accessed on 09/14/2023).

[80] HuggingFace, $Data_config.jsonsentence - transformers/all - mpnet - base - v2atmain - - - huggingface.co$, `https://huggingface.co/sentence-transformers/all-mpnet-base-v2/blob/main/data_config.json`, [Accessed 17-09-2023].

[81] L. McInnes, J. Healy, and J. Melville, *Umap: Uniform manifold approximation and projection for dimension reduction*, 2018. DOI: `10.48550/ARXIV.1802.03426`. [Online]. Available: `https://arxiv.org/abs/1802.03426`.

[82] M. Allaoui, M. L. Kherfi, and A. Cheriet, "Considerably improving clustering algorithms using UMAP dimensionality reduction technique: A comparative study," in *Lecture Notes in Computer Science*, Springer International Publishing, 2020, pp. 317–325. DOI: `10.1007/978-3-030-51935-3_34`. [Online]. Available: `https://doi.org/10.1007/978-3-030-51935-3_34`.

[83] C. Malzer and M. Baum, "A hybrid approach to hierarchical density-based cluster selection," in *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, IEEE, Sep. 2020. DOI: `10.1109/mfi49285.2020.9235263`. [Online]. Available: `https://doi.org/10.1109/mfi49285.2020.9235263`.

[84] HDBSCAN, *Api reference — hdbscan 0.8.1 documentation*, `https://hdbscan.readthedocs.io/en/latest/api.html`, (Accessed on 09/14/2023).

[85] D. Moulavi, P. A. Jaskowiak, R. J. G. B. Campello, A. Zimek, and J. Sander, "Density-based clustering validation," in *Proceedings of the 2014 SIAM International Conference on Data Mining*, Society for Industrial and Applied Mathematics, Apr. 2014. DOI: `10.1137/1.9781611973440.96`. [Online]. Available: `https://doi.org/10.1137/1.9781611973440.96`.

[86] C. E. V. Gallego, V. F. G. Comendador, F. J. S. Nieto, and M. G. Martinez, "Discussion on density-based clustering methods applied for automated identification of airspace flows," in *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, IEEE, Sep. 2018. DOI: 10.1109/dasc.2018.8569219. [Online]. Available: https://doi.org/10.1109/dasc.2018.8569219.

[87] *Parameter selection for hdbscan* — hdbscan 0.8.1 documentation*, https://hdbscan.readthedocs.io/en/latest/parameter_selection.html, (Accessed on 09/14/2023).

[88] hdbscan, *Basic usage of hdbscan* for clustering — hdbscan 0.8.1 documentation*, https://hdbscan.readthedocs.io/en/latest/basic_hdbscan.html, (Accessed on 09/14/2023).

[89] X. Wang and Y. Xu, "An improved index for clustering validation based on silhouette index and calinski-harabasz index," *IOP Conference Series: Materials Science and Engineering*, vol. 569, no. 5, p. 052 024, Jul. 2019. DOI: 10.1088/1757-899x/569/5/052024. [Online]. Available: https://doi.org/10.1088/1757-899x/569/5/052024.

[90] spaCy, *Transformer · spacy api documentation*, https://spacy.io/api/transformer, (Accessed on 09/14/2023).

[91] scikit learn, *Sklearn.preprocessing.labelencoder — scikit-learn 1.3.0 documentation*, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html, (Accessed on 09/15/2023).

[92] A. Hayes, *How stratified random sampling works, with examples*, https://www.investopedia.com/terms/stratified_random_sampling.asp, (Accessed on 09/15/2023), 2023.

[93] B. Ghaddar and J. Naoum-Sawaya, "High dimensional data classification and feature selection using support vector machines," *European Journal of Operational Research*, vol. 265, no. 3, pp. 993–1004, Mar. 2018. DOI: 10.1016/j.ejor.2017.08.040. [Online]. Available: https://doi.org/10.1016/j.ejor.2017.08.040.

[94] N. sharma, *White paper — job skills extraction with lstm and word embeddings — by nikita sharma — medium*, https://nikkisharma536.medium.com/white-paper-job-skills-extraction-with-lstm-and-word-embeddings-d71d1f96024f, (Accessed on 09/15/2023), 2019.

[95] R. Ajayi, *Job skills extraction from data science job posts — by reme ajayi — medium*, https://medium.com/@Olohireme/job-skills-extraction-from-data-science-job-posts-38fd58b94675, (Accessed on 09/15/2023), 2021.

[96] S. Overflow, *Python - what are all possible pos tags of nltk? - stack overflow*, https://stackoverflow.com/questions/15388831/what-are-all-possible-pos-tags-of-nltk, (Accessed on 09/15/2023), 2010.

[97] H. Singh, *Regular expressions — regular expressions in nlp*, https://www.analyticsvidhya.com/blog/2021/03/beginners-guide-to-regular-expressions-in-natural-language-processing/, (Accessed on 09/15/2023), 2022.

[98] https://pypi.org/, *Resume-parser · pypi*, https://pypi.org/project/resume-parser/, (Accessed on 09/15/2023), 2021.

[99] R. Banik, *Hands-On Recommendation Systems with Python*. Packt Publishing, 2018, pp. 46–48, ISBN: 978-1-78899-375-3.

[100] *Trainrev1 — kaggle*, https://www.kaggle.com/datasets/airiddha/trainrev1, (Accessed on 09/15/2023).

[101] *Job search - find 185,543 uk jobs on cv-library*, https://www.cv-library.co.uk/, (Accessed on 09/15/2023).

[102] *Jobs are our job — totaljobs*, https://www.totaljobs.com/, (Accessed on 09/15/2023).

[103] F. Heimerl, S. Lohmann, S. Lange, and T. Ertl, "Word cloud explorer: Text analytics based on word clouds," in *2014 47th Hawaii International Conference on System Sciences*, 2014, pp. 1833–1842. DOI: 10.1109/HICSS.2014.231.

Appendix A

Appendix for Code

A.1 GitHub Link for Code

Table A.1: GitHub Links for Code Segments

Code	**GitHub Link**
Job Description Dataset Preparation and Extraction for Mentioned Categories Code	`https://github.com/shubh2016shiv/thesis-resume-to-job-description-matching/blob/master/Extract%20Job%20Descriptions.ipynb`
Resume Dataset Preparation and Extraction for Mentioned Categories Code	`https://github.com/shubh2016shiv/thesis-resume-to-job-description-matching/blob/master/Extract%20Resume%20Information.ipynb`
Document Clustering Code	`https://github.com/shubh2016shiv/thesis-resume-to-job-description-matching/tree/master/STAGE%20I%20-%20Document%20Clustering`
Document Similarity Code	`https://github.com/shubh2016shiv/thesis-resume-to-job-description-matching/tree/master/STAGE%20II%20-%20Document%20Similarity`
Skill Extraction Pipeline Code	`https://github.com/shubh2016shiv/thesis-resume-to-job-description-matching/tree/master/STAGE%20III%20-%20Skill%20Extraction`
Resume Matching to Job Description Pipeline Code	`https://github.com/shubh2016shiv/thesis-resume-to-job-description-matching/blob/master/resume/matching_score_between_resume_and_JD.ipynb`

Appendix B

Appendix for No-SQL MongoDB Database

B.1 Job Descriptions stored on MongoDB Database

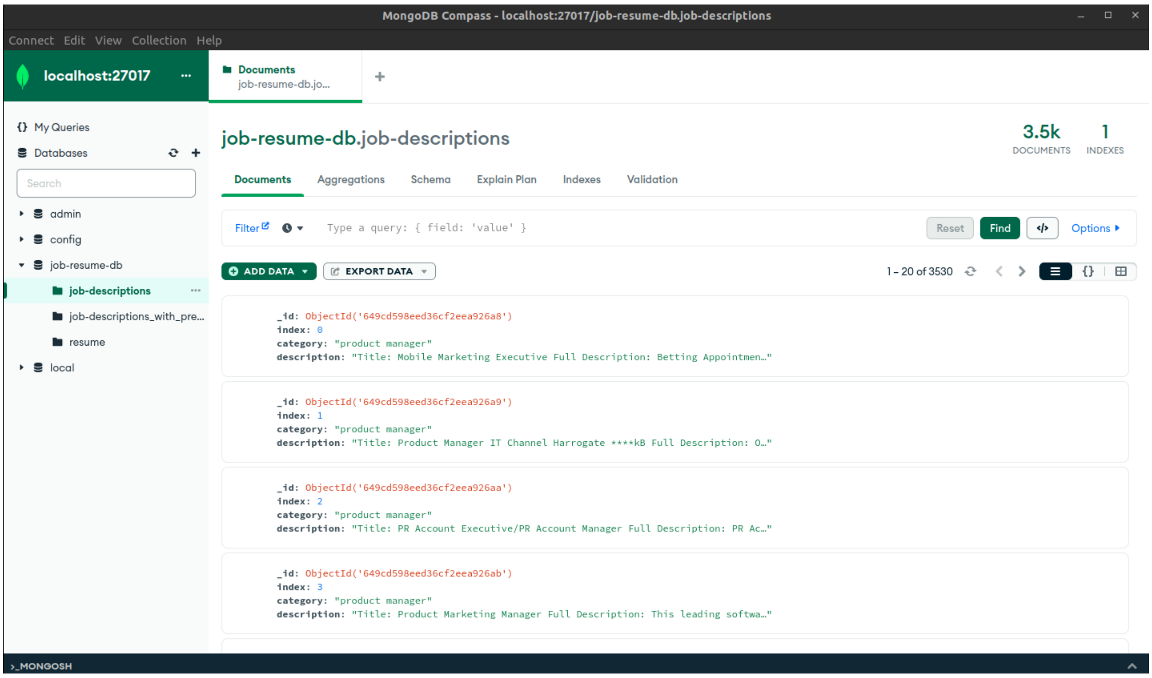


Figure B.1: Job Description stored in MongoDB Database inside the collection called `job-descriptions`

B.2 Resume stored on MongoDB Database

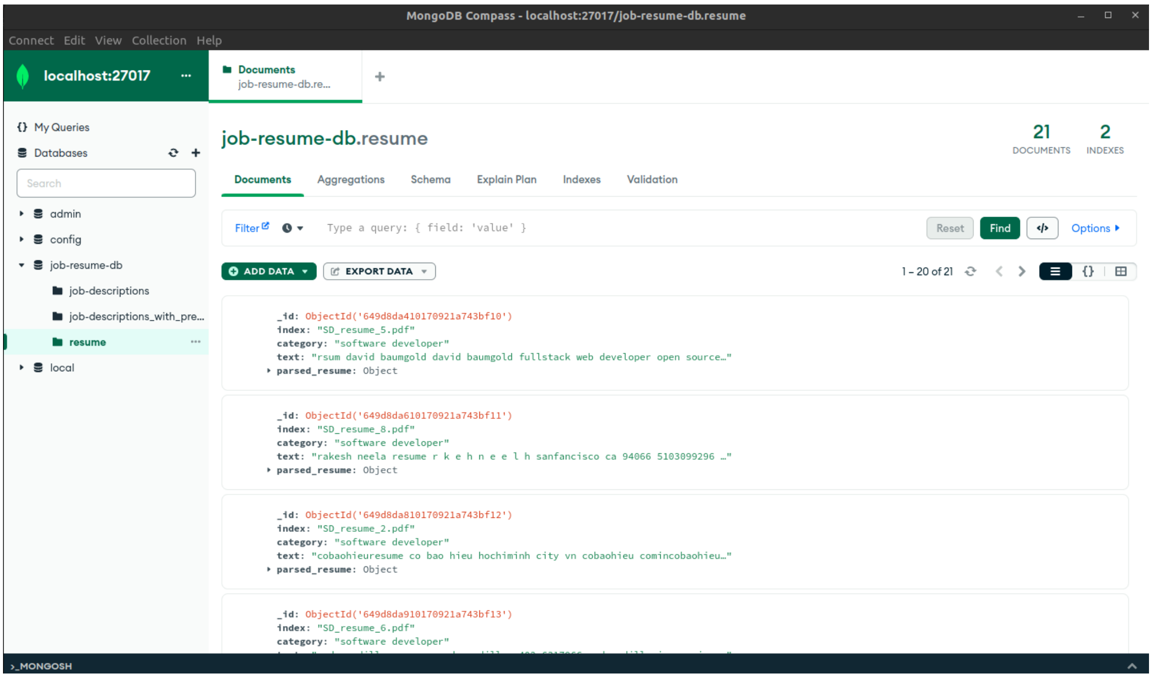


Figure B.2: Resume stored in MongoDB Database inside the collection called `resume`

Appendix C

Extracted Noise Keywords, 100 Least Common Words, 100 Uni-Gram and Bi-Gram Words

Below are the noisy keywords, 100 least common words, 100 uni-gram and bi-gram which were found either using the manual inspection or visual analysis:

Table C.1: GitHub Links for Additional Resources

Resource	**GitHub Link**
NOISY KEYWORDS	`https://github.com/shubh2016shiv/thesis-resume-to-job-description-matching/blob/master/STAGE%20II%20-%20Document%20Similarity/noisy_words.txt`
100 LEAST COMMON WORDS	`https://github.com/shubh2016shiv/thesis-resume-to-job-description-matching/blob/master/STAGE%20II%20-%20Document%20Similarity/least_common.txt`
BOTTOM 100 UNI-GRAM WORDS	`https://github.com/shubh2016shiv/thesis-resume-to-job-description-matching/blob/master/STAGE%20II%20-%20Document%20Similarity/bottom_100_unigram_words.txt`
BOTTOM 100 BI-GRAM WORDS	`https://github.com/shubh2016shiv/thesis-resume-to-job-description-matching/blob/master/STAGE%20II%20-%20Document%20Similarity/bottom_100_bigram_words.txt`