# Job clustering: An unsupervised approach for a recommender system of skill requirements

Thien Binh Hoang Dao

Student number: 2049124

Thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Data Science & Society

Department of Cognitive Science & Artificial Intelligence

School of Humanities and Digital Sciences

Tilburg University

Thesis Committee:

Dr. Peter Hendrix

Tilburg University

School of Humanities and Digital Sciences

Department of Cognitive Science & Artificial Intelligence

Tilburg, The Netherlands

May 17, 2021

## Contents

## List of Figures

## List of Tables

## List of Acronyms

**AI**

Artificial Intelligence

**ARTM**

Additive Regularization of Topic Model

**BERT**

Bidirectional Encoder Representations from Transformers

**BIRCH**

Balanced Iterative Reducing and Clustering using Hierarchies

**DISCO**

European Dictionary of Skills and Competences

**GSDMM**

Gibbs Sampling Dirichlet Mixture Model

**HDBSCAN**

Hierarchical Density-Based Spatial Clustering of Applications with Noise

**LDA**

Latent Dirichlet Allocation

**ML**

Machine Learning

**PCA**

Principal Component Analysis

**PPA**

Post-Processing Algorithm

**RAKE**

Rapid Automatic Keyword Extraction

**RF**

Random Forest

**TC**

Term Contribution

**TF-IDF**

Term Frequency-Inverse Document Frequency

**TV**

Term Variance

**URF**

Unsupervised Random Forest

**YAKE!**

Yet Another Keyword Extractor

## Introduction

From a survey conducted by Pew Research Center in 2015, 54% of Americans use Internet for their job searches. Among the U.S. job seekers during 2014-2015, 79% utilized online information to seek new jobs, and 34% of them considered it the most important resource (Smith, 2015). Indeed, online job advertisements seem to be a good source of reflection on actual skills required to pursue a career. However, those who have a low education level found it difficult or experienced a lack of confidence in going online to look for new jobs (Smith, 2015). Besides, job seekers may have to invest a lot of time and effort in typing, clicking, scrolling, skimming, and scanning through many online career portals to figure out what skills they need to equip themselves with or what requirements they need to satisfy. Thanks to the support of data science algorithms, it is possible to automatize this task. This thesis project aims to build an automatic model based on clustering algorithms to recommend the skill set required for job seekers.

With the emergence of Machine Learning (ML), and Artificial Intelligence (AI), new careers have been introduced. By 2025, it is estimated by World Economic Forum (2020) that there may be 97 million new job roles appearing across 26 economies and 15 industries. Nevertheless, those new jobs may have different titles in different companies. That makes job seekers confused, especially the inexperienced ones, who may even not be aware of the existence of new careers. It would be better if the job is recommended not only by the job title but also by the skill set required. The recommender proposed in this thesis project would considerably help young people who may not know the exact title of the jobs they have strengths or passions on.

Besides, together with the appearance of new careers, there are also the elimination or obsolescence of several jobs, as a consequence of human-machine labor shifting. For example, data entry clerks or assembly workers may disappear, giving way to faster and more efficient machines and robots (World Economic Forum, 2020). The employees who are working in those fields may be urged to seek new job to ensure their career paths. Those kind of job seekers would like to switch to the jobs requiring as

least new skills they need to equip as possible. As a result, the time spent to learning new skills can reduce and the risk of a salary loss can be avoided. Thus, the recommender built from this thesis project should be able to satisfy this need.

Specifically, the main research question addressed in this thesis project is as follows:

- **RQ**: Is it possible to generate skill set recommendation for job seekers with the use of machine learning methods?

The main research question is answered through exploring the following sub-questions:

- **SQ1**: What information in the online job posts is most useful to build an automatic recommender system of skill set?
- **SQ2**: In the context of skill set recommendation, which clustering method yields the best performance?

The recommender system of jobs and skills based on online job postings has been addressed in the literature. However, most of the models proposed in the literature are operated semi-automatically, which means the presence of annotators or target labels is needed (Calanca, Sayfullina, Minkus, Wagner, & Malmi, 2019; Dave, Zhang, Al Hasan, AlJadda, & Korayem, 2018; De Mauro, Greco, Grimaldi, & Ritala, 2018; Djumalieva, Lima, & Sleeman, 2018; Mhamdi, Moulouki, El Ghoumari, Azzouazi, & Moussaid, 2020). Obviously, not every data set has target labels available, and the process of label marking requires considerable time and efforts. As a consequence, the mentioned models are impractical in most of the cases. Thus, this thesis project seeks to find an unsupervised approach in building a recommender system to avoid the huge cost of label annotation.

Taking a look at the existing literature, one may realize that job title and job description are the most common information used to build a model of classification (Boselli, Cesarini, Mercorio, & Mezzanzanica, 2018) or clustering (Djumalieva et al.,

2018; Mhamdi et al., 2020). However, in the context of recommending skill sets instead of jobs only, it may be inadequate to only rely on this information as input to the machine learning models. This thesis research aims to find out which is the most useful information to feed into the clustering models.

To build a recommender of skill sets, Mhamdi et al. (2020) theoretically proposed using word2vec with k-means clustering. Meanwhile, Vinel, Ryazanov, Botov, and Nikolaev (2019) experimented with text representation such as Term Frequency-Inverse Document Frequency (TF-IDF), word2vec, Bidirectional Encoder Representations from Transformers (BERT), and Additive Regularization of Topic Model (ARTM). Combining with several clustering methods - including k-means, Affinity Propagation, Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH), Agglomerative clustering, and Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN), Vinel et al. (2019) concluded that k-means+ARTM and BIRCH+ARTM were the most applicable combinations to consider. However, in order to evaluate the performance of different models, they used a manually pre-marked set of job specializations. The manual marked list may become outdated or inadequate over time. The evaluation of the recommender system may therefore not reflect the performance of the algorithm in reality. Hence, this thesis project aims to eliminate the utilization of all manual work during clustering performance evaluation.

The results from this thesis project implied that ....

## Related work

A recommender system is a program aiming to filter information and seek the prediction of a user's preference to recommend top items related to their interests (Lu, Wu, Mao, Wang, & Zhang, 2015). Building a recommender system of jobs and skill sets can be done based on solving the problem of text clustering (Lu et al., 2015). This section will review the existing literature on text clustering in general, and job clustering in particular.

Generally, a text clustering project can be approached from two directions: semi-supervised or unsupervised learning. While semi-supervised approach partially

needs the involvement of external resources to train the model, unsupervised approach only utilizes the dataset itself to learn the patterns. However, because the data used in text clustering usually are documents containing thousands, millions, or even billions of words, it implies "the curse of dimensionality". According to a review of Alelyani, Tang, and Liu (2013), this issue does not only lead to model complexity and computation cost, but also overfitting and decreasing performance. Thus, dimensionality reduction techniques such as feature selection or feature extraction need to be done before applying clustering of either approach, semi-supervised or unsupervised learning.

## Dimensionality reduction

The task of reducing dimensions can be done by feature selection (selecting a smaller set of variables from original variables), feature extraction (selecting and combining original variables into features), or the combination of both. For feature selection, the smaller number of variables that satisfies a given criterion are selected from original variables. Meanwhile, feature extraction methods seek to combine several variables to produce features; thus, no original variable can be observed in the new feature set.

### *Feature selection*

As categorized by H. Liu and Yu (2005), features can be selected via ranking or subsetting. For rank-based feature selection techniques, the features are ranked based on their importance then only the ones with highest ranks are kept. Meanwhile, minimum subset algorithms seek to find out the subset of as least features as possible. Feature subset selection may not be preferred in dealing with large dataset because it can be time consuming in order to search for the minimum subset of features (Alelyani et al., 2013). Therefore, feature ranking techniques seem to be more popular in text clustering. Among rank-based feature selection methods, the common ones are TF-IDF reviewed by L. Liu, Kang, Yu, and Wang (2005), Term Contribution (TC) introduced by T. Liu, Liu, Chen, and Ma (2003), and Term Variance (TV) proposed by L. Liu et al. (2005). As expressed in their names, those methods focus on sorting terms based on a pre-defined formula and only top terms are selected as input to machine learning

models.

Furthermore, keyword extraction is also a common approach applied in text-related tasks to reduce the number of words to handle. Different with term-based methods mentioned above, keyword extraction methods return not only single terms but also phrases. Among the variety of keyword extraction techniques, TextRank (Mihalcea & Tarau, 2004), PositionRank (Florescu & Caragea, 2017), RAKE (Rose, Engel, Cramer, & Cowley, 2010), or Yet Another Keyword Extractor (YAKE!) (Campos et al., 2020) are the state-of-the-art ones. Based on several criteria such as word frequency, word length, or word co-occurence, each word is scored by the algorithm and the returned output is a list of words or phrases with the highest scores, which is called keywords or keyphrases. This approach is useful for automated tasks such as text summarizing, tagging, or recommender building. However, when handling large text documents, one may notice that TextRank and PositionRank are rather low in computing keywords' scores, which may influence the whole process, especially under the circumstances of time constraint.

***Feature extraction***

In spite of the reputation of the mentioned term-related techniques, they may fail in the tasks that require to analyze the semantic of words. Hence, the feature extraction algorithms attempt to generate text embeddings, which are the numerical representation of the text as well as its semantic properties. In the embedding space, similar words will have similar representation and therefore stay close to each other, and vice versa. In the experiment of job clustering comparing TF-IDF and word embeddings, Vinel et al. (2019) found that the cluster models performed better with word embeddings. Calanca et al. (2019), and Mhamdi et al. (2020) also applied word embeddings in their research of job recommendation.

However, word embeddings, the smallest unit representing words in the embedding space, usually are numerical vectors with length between 100 and 300, or 768 in some special models. That leads to challenges of memory capacity, especially with the document sets containing a variety of unique words (Raunak, Gupta, & Metze,

2019). For that reason, researchers tend to combine word embeddings with other feature extraction methods such as PCA to reduce the dimensions of word embeddings. PCA is widely used not only because of its simplicity but also its fast computation. Raunak et al. (2019) also proposed using PCA together with post-processing algorithms on the word embedding to both shrink the embedding size to half and improve the quality of the reduced embedding.

One other method that deserves mentioning is URF although it has not been widely researched. The introduction of URF came from the idea that adding a set of "synthetic" data into the "original" data, one can train the supervised Random Forest (RF) as normal because the problem now is turned to binary classification between "synthetic" and "original" (Breiman, 2002). Elghazel and Aussem (2015) commented in their paper that URF can handle missing variables, detect outliers, and most importantly, the variable importance obtained from URF can be used in selecting features.

Researches on job clustering in general and skill set recommendation in particular are very limited in the literature, especially in term of feature selection. Most of papers only stated that job titles, job descriptions, or the whole job ads are input to their models. Meanwhile, the job postings have other sections such as job requirements, qualification required, which are all have potentiality to become features for skill set clustering. Thus, this thesis, through obtaining URF's variable importance, attempts to figure out which sections are useful information for clustering models.

## Clustering approaches

### *Semi-supervised learning*

According to Grira, Crucianu, and Boujemaa (2005), the clustering process can be conducted with the guidance or adjustment from external knowledge like class labels. This approach is called semi-supervised learning. Those independent labels can play a role as seed sets, or centroids to initialize the clustering (Qin, Ding, Wang, & Wang, 2019). In other words, there are partial labelled data in the whole dataset and semi-supervised clustering methods will attempt to cluster the rest of dataset based on

the existing labelled data. For example, Djumalieva et al. (2018) grouped the skills based on the similarity with the taxonomy of the ONS 2010 Index (Office for National Statistics) and the European Dictionary of Skills and Competences (DISCO). The advantage of this approach is the ability to take use of prior knowledge, but it fails to handle large scaled dataset (Reddy, Viswanath, & Reddy, 2018).

A second method that can be considered semi-supervised is a two-phase clustering process, which involves supervised learning in one phase and unsupervised learning in the other. For instance, De Mauro et al. (2018) consulted domain knowledge experts to build a list of job families before applying unsupervised clustering to identify the skill sets. Similarly, Calanca et al. (2019) obtained the opinions of experts in combination with crowd-sourcing to annotate the collection of soft skills before using them to extract and parse similar skill phrases from online job advertisements. Obviously, not every data set has the target labels available, and the process of label marking seems to consume efforts and time. As a consequence, the mentioned models for semi-supervised learning is time-consuming and impractical in most cases.

***Unsupervised learning***

Unsupervised learning is the traditional approach of clustering, where labelled data are not available. Among the variety methods of unsupervised clustering, k-means is the most popular one due to its simplicity. For instance, Mhamdi et al. (2020) proposed an unsupervised model to cluster the job title and description together with the behaviors of job seekers as a recommender system. They theoretically described a model based on k-means algorithm. However, they did not use the skill requirements as features in the model.

Hierarchical clustering and density based clustering are also well-established methods used in text clustering (Suyal, Panwar, & Singh Negi, 2014). Genie is a representative technique for hierarchical clustering and HDBSCAN is an example of a density based clustering technique. While k-means method requires the input data to be in numeric format and to have a convex shape, Genie is not constrained by the input format and still produces high accuracy. Besides, Gagolewski, Bartoszuk, and Cena

(2016) emphasize that Genie runs very fast and consumes less memory, which makes it suitable for dealing with large data sets. However, Suyal et al. (2014) indicated that sometimes it is difficult to decide to "cut the tree", which means finalizing the number of clusters found, and this task may need the support from domain knowledge experts. Therefore, Campello, Moulavi, and Sander (2013) proposed a "density-based cluster hierarchy" method and called it HDBSCAN. As described in their papers, clusters generated by HDBSCAN are grouped from dense regions, which are separated by sparse regions, and the hierarchy tree is constructed on the basis that clusters of higher density level will be on top of those in lower density areas. There is no need to define the number of clusters a priori, which overcomes the difficulty occurring in normal hierarchical clustering. Moreover, HDBSCAN can help detect outliers or noises. Nevertheless, when applying in job clustering, Vinel et al. (2019) had to stop applying HDBSCAN algorithm due to its low quality score and excessive amount of noise detection.

Another well-known research direction in unsupervised text clustering worth mentioning is topic modelling. Latent Dirichlet Allocation (LDA) is an example of this sub-class of clustering. As implied by Sun (2014), LDA solves the problem of semantic structure being ignored in keyword models like k-means. In addition, LDA creates a soft clustering, where one text document can belong to more than one cluster, depending on its probabilities related to possible topics. Empirically, Poch, Bel Rafecas, Espeja, and Navio (2014) applied an LDA variant to construct clusters of job offers before ranking them to serve the job seekers. The results of their study revealed superior performance of LDA as compared to alternative clustering techniques such as k-means. In spite of its advantages, LDA seems to not perform well in the context of clustering short text such as job titles, where the frequency of words in documents is typically 1. Hence, the document should be mapped to only one topic (Qiang, Qian, Li, Yuan, & Wu, 2020). Thus, Yin and Wang (2014) suggested the Gibbs Sampling Dirichlet Mixture Model (GSDMM), where each document is assigned to only one topic and the number of topics is decided automatically by the algorithm with fast convergence.

Although there have been several papers seeking to conduct job clustering, a comparison of the discussed clustering techniques in the context of skill set recommendation systems has not yet existed in the literature. Thus, this thesis project aims to compare clustering performance of different models with a single dataset. Specifically, the models applied in this thesis are k-means, Genie, HDBSCAN, LDA, and GSDMM. Results from the comparison will provide more insights into the suitability of different clustering algorithms in building recommender systems of skill sets.

## Methods

In this section, each of dimensionality reduction techniques used in this research is described, namely: RAKE keyword extraction, a series of PPA plus PCA, and finally URF. Besides, the underlying mechanisms of selected clustering models are also presented together with motivation.

### Dimensionality reduction

#### *RAKE*

RAKE technique was applied to extract keyphrases from long text documents (Rose et al., 2010). In details, RAKE splitted the text based on delimiters such as punctuations or stopwords into candidate keyphrases. For each candidate keyphrase ($w$), the score is computed for each member word then sum up based on one of the following metrics:

- $freq(w)$: word frequency, in which words that occurs more in the list of candidate keyphrases will score higher.
- $deg(w)$: word degree, in which words that both occurs in longer candidate keyphrases and co-occurs more in the list of candidate keyphrases will score higher.
- $deg(w)/freq(w)$: the ratio of word degree to frequency, in which words that occur very often (high $freq$) but not in long candidates (low $deg$) will have less scores, and vice versa.

### *PPA and PCA*

After extracting the desired list of words, each word was mapped with a pre-trained word embedding, then the word embedding matrix was operated on PPA and PCA to reduce the dimension to half size but still maintain the quality of word representation, as described in Raunak et al. (2019). Specifically, a pipeline of PPA, PCA, and PPA was conducted. PCA is a statistical technique that seeks to obtain the new set of variables (components) that best reflecting the data variation. Meanwhile, PPA is a series of vector operation done on word embeddings. Details of the pipeline are as follows:

1. PPA: Subtracting the mean from word embeddings to obtain temporary word embeddings, operating PCA on those temporary embeddings to figure out top-$D$ components ($D$ used in the original paper is 7) among $N$ original dimensions, then removing those top-$D$ components to obtain post-processed embeddings.

2. PCA: Operating PCA with the embeddings yielded from Step 1 to obtain the reduced embeddings with $N/2$ dimensions.

3. PPA: Conducting similar post-processing operations in Step 1 with the reduced embeddings yielded from Step 2.

### *URF*

The idea of feature selection is done through the feature importance computed from training a URF classifier. In general, URF works similarly to the traditional RF algorithm, which learns from a large number of decision trees, showing probabilities of all outcomes depending on different conditions of features (see Figure 1 for illustration of a decision tree). However, RF is trained on a labelled data to solve classification or regression problems, whereas URF needs to operate without target labels. Breiman (2002) proposed to create a subset of new data following a random distribution from the existing dataset, label each subset target labels of "original" and "synthetic", merge both subsets into one dataset then feed into the RF classifier. This approach is called

URF. According to Breiman (2002), the existence of the synthetic data breaks the dependency of features (if any) in the original dataset. As a result, an unsupervised learning problem is turned into a binary classification. Therefore, based on the scores of feature importance obtained from the trained RF classifier, one can decide to omit less important features.

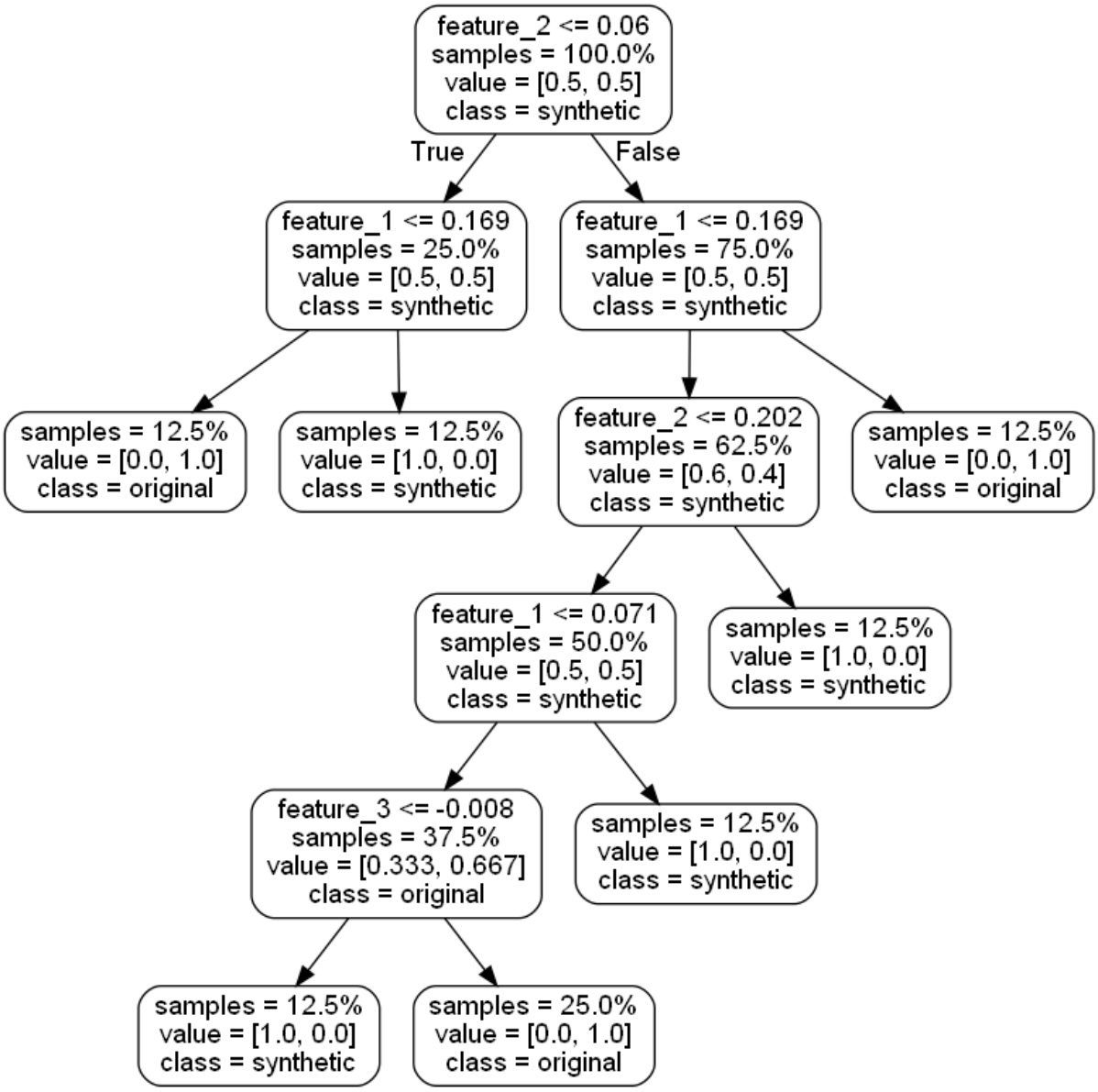


*Figure 1*. Illustration of Decision Tree

## Clustering

Clustering is described as the task of gathering similar data points into groups such that data points belonging to the same cluster stay close together and separate

from other clusters. Depending on the purpose of study, the measure of similarity between data points may be different; thus, the clustering results may be different in various cases (Grira et al., 2005).

The clustering algorithms applied in this research are k-means (partition-based), Genie (hierarchical-based), HDBSCAN (density-based), LDA, and GSDMM (probability-based).

### *k-means*

k-means clustering works under a simple mechanism, where users pre-specify hyper parameter k, which is the desired number of clusters (Han & Kamber, 2012). At the first step, k random centroids (the central point of clusters) are selected. At the second step, each of remain points is assigned to the most similar cluster, which means the closest centroid. The third step is done after all points are clustered, a new centroid of the cluster is computed by taking the mean of all points assigned to that cluster. The algorithm will update the new means and assign points to clusters based on those new centroids as in step 2. Step 3 will be repeated afterward until there is no change in centroids, which means the clusters become stable.

Due to its simplicity, fast computation, and ability to deal with various data types, k-means is selected as a baseline in this research.

### *Genie*

The second clustering method selected for this research is Genie, a fast hierarchical-based clustering method, recently proposed by Gagolewski et al. (2016). Hierarchy clustering allows to divide a cluster into subgroups, or "nested" clusters, which construct a hierarchy tree of clusters. It is different with a "flatten" clustering like partitioning methods. Besides, Genie's performance is proved to be unaffected by outliers.

A normal hierarchy cluster tree can be built from bottom up, which means each data point from the beginning is considered as a single cluster, and the pair of data points from two clusters with minimum distance will be merged into one larger cluster. The iteration of merging continues with other pairs of the remain clusters until no pair

left unmerged. That indicates the finish of the bottom layer of the tree. The second and higher layers are also constructed by the same criterion: clusters that have the closest pair of data points will be merged. When the top layer only has one large cluster, the tree is completely built.

However, a Genie cluster tree has one difference in merging criterion to ensure the sizes are not so unequal between clusters, which leads to its advantage of being less sensitive to outliers. The merging criterion is based on a threshold of an economic inequity measure (the Gini index, for instance): smaller clusters are forced to merge if the Gini index of those cluster sizes is higher than the threshold. Thus, the Gini index is updated after each operation of merging (see Equation (1) for incremental computation of Gini index). That is also the reason why Genie is considered a fast computation method, with $O(n)$ operations rather than $O(n^2)$ like other traditional hierarchical clustering methods.

At the beginning, every cluster has the same size of 1, thus, the Gini index is 0. If denoting $g_j$ as the Gini index value at the time when there are $n-j$ clusters, $j = 0, 1, ..., n-1$, $g_j$ will be computed as in Equation (1), where $c_{s_1}$ and $c_{s_2}$ are the cluster sizes obtained when operating clustering from step $j-1$ to $j$.

$$g_j = \frac{(n-j)ng_{j-1} + \sum_{i=1}^{n-j+1}(|c_i - c_{s_1} - c_{s_2}| - |c_i - c_{s_1}| - |c_i - c_{s_2}|) - c_{s_2} - c_{s_1} + |c_{s_1} - c_{s_2}|}{(n-j-1)n} \tag{1}$$

***HDBSCAN***

HDBSCAN, introduced by Campello et al. (2013), is a density-based clustering algorithm, which means the clustering is operated based on the density of the data points: the denser the area is, the higher ability a cluster is formed. However, unlike Genie and k-means, which are complete clustering methods with all points belonging to one cluster, HDBSCAN will ignore sparse regions and marked them as noises during the process of clustering. This is also the reason why HDBSCAN is called the outlier detector.

Next, one may ask how HDBSCAN can estimate the density of the points to

make decision of forming clusters or marking noises. This estimation will be done indirectly through the user pre-set minimum samples hyperparameter. For example, the minimum samples is set as 7, HDBSCAN will compute the furthest distance a point has to travel to reach its closest 7 points. That distance is called core distance, and density is estimated by taking the inverse of core distance. It can be easily observed that smaller core distances indicate denser areas and vice versa, sparser regions contains points with larger core distances.

After computing the core distances of all points, the mutual reachability distances between each pair of points need to be obtained using Equation (2)

$$d_{mreach-k}(a-b) = max\{core_k(a), core_k(b), d(a,b)\} \tag{2}$$

where $core_k(a)$ and $core_k(b)$ are the core distance for $k$ min samples of point $a$ and point $b$, respectively; $d(a,b)$ is the original distance between point $a$ and point $b$.

Unlike Genie method, an HDBSCAN cluster tree is formed from top to bottom, which means the splitting will begin from a single large cluster. Next, with a threshold value ($\lambda$), if any mutual reachability distance is higher than the threshold, which means the points are not close enough and not in the dense region, the tree is splitted into smaller clusters. Gradually decrease the threshold value, the splitting operation continue until the threshold cannot be lower. Finally, a complete cluster tree based on density levels is constructed.

Moreover, HDBSCAN allows users to decide which clusters are important enough to keep based on the hyperparameter minimum cluster size. If during the process of splitting the clusster tree, any cluster smaller than the minimum cluster size will be dropped out of the final cluster tree. This will help smoothing the density probability to reach the real cluster.

Instead of pre-defining number of clusters like k-means, HDBSCAN will return the number of clusters automatically from the internal working mechanism. The only hyper parameters needed to feed into HDBSCAN model are the minimum samples and the minimum cluster size. Besides, it is fast, which makes it suitable to deal with large

dataset, such as this thesis project.

### *LDA*

LDA is a statistical modelling method proposed by Blei, Ng, and Jordan (2003). LDA is selected for this research because it can handle text data directly, without the need to use numerical representation such as text embeddings like other methods discussed above (k-means, Genie, and HDBSCAN).

LDA belongs to probability-based clustering method because it works based on assumption of probability distribution: each document is a distribution over topics, and each topic is a distribution over words. In other words, one document can be assigned more than one topic, and in one topic, some words may show up more than other words. The number of topics $N$ is pre-defined by user.

Next, to generate the multinomial probabilities of topics for each document and of words per topic, assuming those probabilities have Dirichlet prior distributions, LDA needs the user to set two Dirichlet priors: alpha and beta. Alpha controls the density of topics per document, which means higher alpha leads to more topics assigned to each document while lower alpha means a document consist only few topic. Meanwhile, beta determines the distribution of words per topic, lower beta means few words per topic and vice versa, higher beta leads to many words contained in each topic. Generally, it is expected that each document should have not many topics assigned and only a few words representing each topic. Thus, alpha and beta should be set at values less than 1.

### *GSDMM*

GSDMM is short for the Dirichlet Multinomial Mixture model using collapsed Gibbs Sampling algorithm, proposed by Yin and Wang (2014). Similar to LDA, GSDMM is also based on probability of topic distribution per document and word distribution per topic. However, each document can only consist one topic, which makes it suitable for short text clustering. This is also the reason that GSDMM is selected for this thesis project. Besides, there is no need to pre-set the number of topics, because the algorithm will do that task by itself. Instead, the maximum number of possible topics, $K$, need to be pre-defined by user. The authors advised that $K$ should be set

higher than the expected number of topics, because the final assigned topics may be less than $K$.

GSDMM process starts with randomly assigning the documents to $K$ topics. Afterward, the documents are reassigned to the other topics, following two rules:

- Rule 1: The document will be reassigned to the topic with more documents.
- Rule 2: The documents belonging to the reassigned topic contain more same words with the current document.

The loop continues until it reaches the max iterations. Finally, one can observe that some topics consist several documents and some topics are empty, and the documents of the same topic contain similar words. Similar to LDA, GSDMM also needs to pre-define alpha and beta. In the case of GSDMM, alpha influences Rule 1 and beta impacts Rule 2. The authors of GSDMM used an analogy to describe that process, namely the Movie Group Process, where topics are tables, documents are students, and words are the movies.

## Experiment Setup

### System configuration

The thesis project was hosted on a Windows 10 laptop, equiped with Intel i5 core, 512GB SSD, RAM 8GB and NVIDIA GeForce MX250. Most of the tasks were processed by R 4.0.0 and python 3.8.5. The variety of R libraries and python packages were applied, all are stated in the following subsections.

### Dataset

The dataset used in the thesis project can be accessed from `https://kaggle.com/udacity/armenian-online-job-postings`, naming Armenian Online Job Postings, which was compiled by Datamotus CEO. The initial dataset contains 19,001 online job advertisements retrieved from website `https://careercenter.am/` (an Armenian career portal) during 2004-2015.

With the purpose of this research, only four columns of the initial dataset are needed, namely *Title* (job title), *JobDescription* (job description), *JobRequirment* (job

requirements), and *RequiredQual* (required qualifications) (see Table 1). Column *jobpost* only served for purpose of missing value treatment and was omitted before carrying out the text-preprocessing steps.

Table 1

*Ininitial dataset description*

| Column | Data type | Observations | Unique values |
|---|---|---|---|
| jobpost | character | 19001 | 18892 |
| Title | character | 18973 | 8636 |
| JobDescription | character | 15110 | 12862 |
| JobRequirment | character | 16479 | 14182 |
| RequiredQual | character | 18517 | 16688 |

**Missing value treatment**

All treatments of missing values were done in R using the following packages: dplyr, stringr, purrr, and Hmisc for data exploration and manipulation; naniar, and ggplot2 for data visualization.

All values of "NA", "N/A", "na", "NaN", "", and default NA of R program are considered missing values. There were many missing values in the initial dataset (see Figure 2), the first treatment was made to impute missing values. By using regular expressions, the attempt was to extract text from corresponding sections of column *jobpost* then impute into corresponding columns *Title*, *JobDescription*, *JobRequirment*, and *RequiredQual* (see Figure 3 of parsing result).

The next treatment was omitting 7 rows that have missing values of all 4 columns and 2 rows with missing values of 3 columns because those rows could not be used in clustering. The remain dataset consists of 18992 rows, where all missing values were imputed with the word "Unprovided" to reflect the reality that job recruiters indeed did not provide information of corresponding sections in the job posts.

The final dataset has one more column *job_id*, which is the row number in the

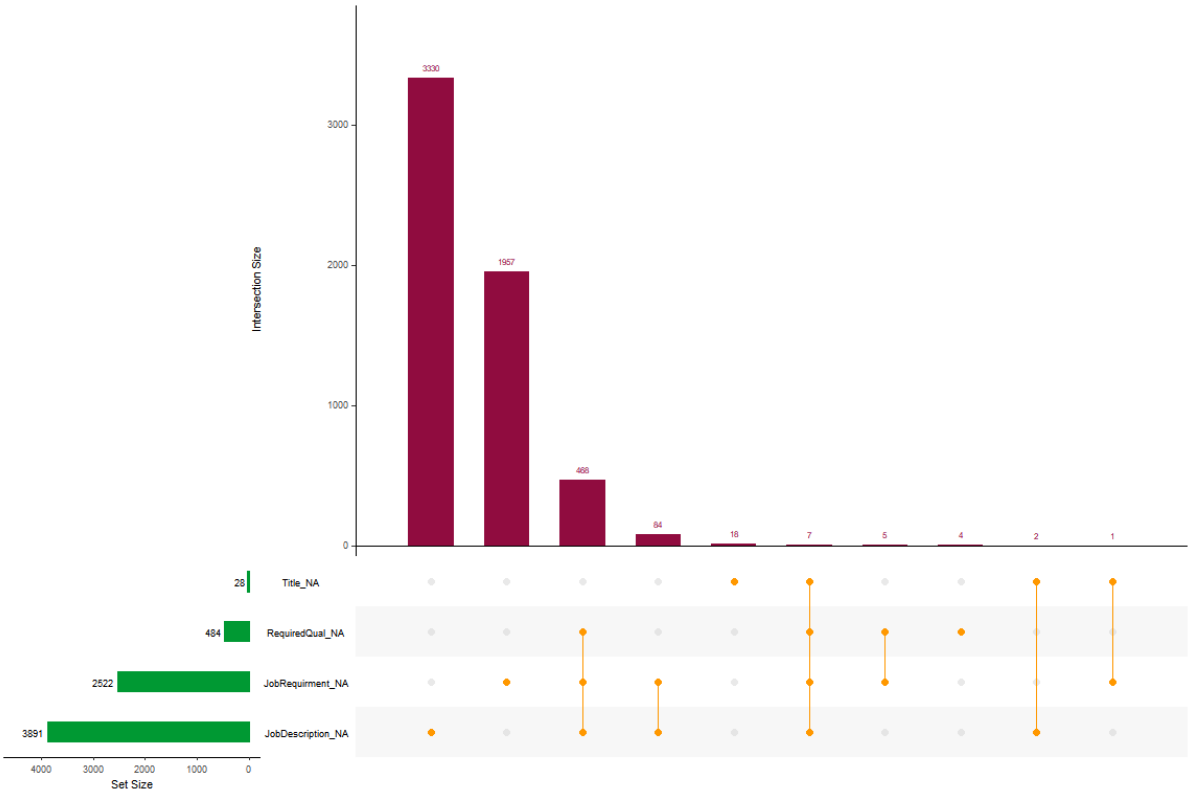


*Figure 2*. Missingness of initial dataset. Data sources: `https://kaggle.com/udacity/armenian-online-job-postings`

original dataset, for later use of mapping job titles and skill sets. The remain columns were also renamed for convenience and consistency in later usage (see Table 2).

Table 2

*Final dataset description*

| Column | Old column name | Observations | Unique values | Imputed values |
|---|---|---|---|---|
| job_id | row number | 18992 | 18992 | 0 |
| job_title | Title | 18992 | 8648 | 0 |
| job_description | JobDescription | 18992 | 13204 | 3409 |
| job_requirement | JobRequirment | 18992 | 14188 | 2508 |
| job_qualification | RequiredQual | 18992 | 16692 | 472 |

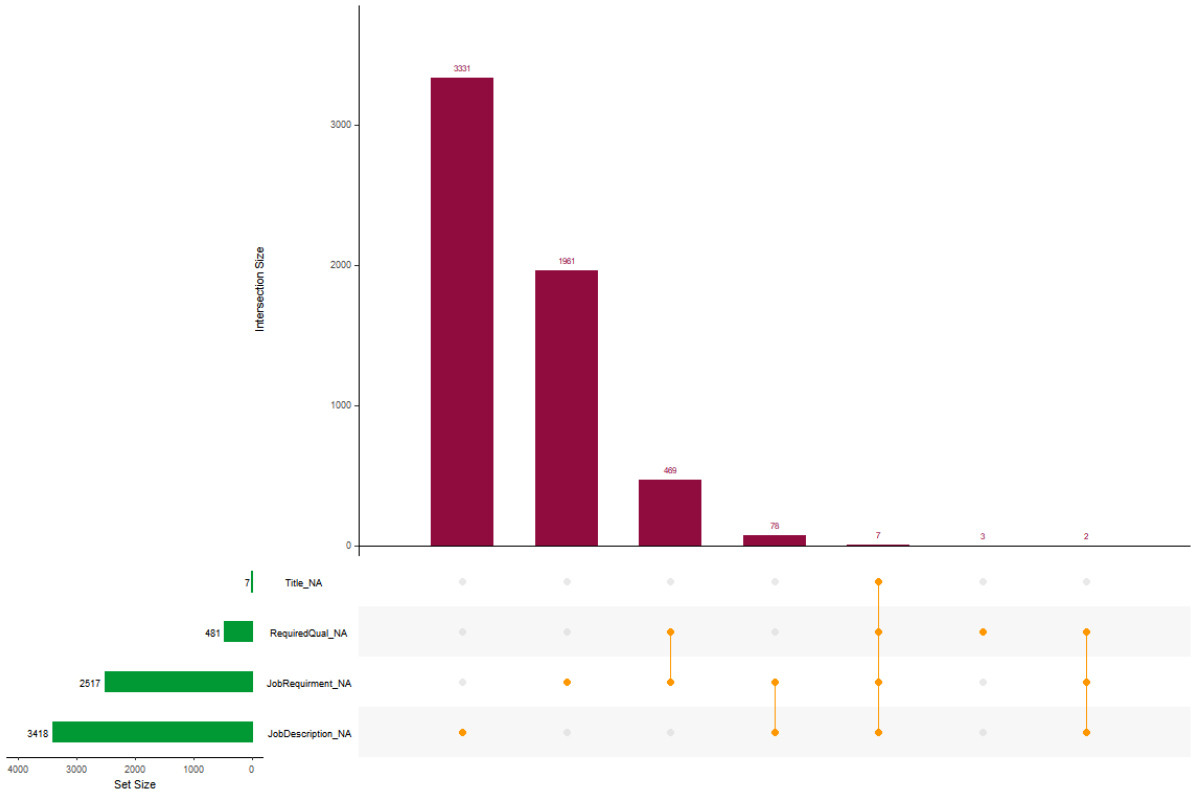


*Figure 3*. Missingness of dataset after extracting and parsing text from column *jobpost*

### Text pre-processing

All the text pre-processing tasks were done by python, using the following packages: numpy, pandas, re, string, nltk, and rake_nltk.

For job titles, the text in column *job_title* of the dataset was pre-processed following this pipeline: lower casing, URL removal, removal of special characters (including punctuations, numbers, and line break characters), word tokenizing, removal of English stopwords, and lemmatizing. The job title subset contains only *job_id*, original *job_title*, and pre-processed *job_title*.

For skill sets, a different pre-process was applied to three columns *job_description*, *job_requirement*, and *job_qualification*: lower casing, URL removal, replacing all special characters (including punctuations, numbers, and line break characters) with character "|", extracting top 10 key phrases with length from 1 to 5 words per phrase (using RAKE algorithm), removal of English stopwords, removal of character "|", word tokenizing, and lemmatizing. Afterward, the dataset was reshaped

to long format such that one row only consists one skill phrase. The skill subset contains only *job_id*, *skill_id* (row number per *job_id*), *column_type* bearing one of the following values "job_description", "job_requirement", and "job_qualification", the original *skill* before lemmatizing (serving for later use of visualization), and *skill_lemma* containing the lemmatized skill phrases (serving for later use of mapping to word embeddings).

**Text embeddings**

For LDA and GSDMM, there is no need to further process the text because those algorithms operate directly with text data. Nevertheless, k-means, Genie, and HDBSCAN only use numeric data as input. Thus, it was required to convert the pre-processed text data to text embeddings. In order to make effective use of words' semantic properties, this research applied the pre-trained word embeddings, namely *en_core_web_lg*, available to download through the python package *spaCy*. The original loaded word embeddings have 685,000 keys and 685,000 unique vectors with length of 300 dimensions.

The embeddings for job titles, which will be called title embeddings from now on, were generated by taking average of all L2 normalized word embeddings per each job title. Thus, the final title embeddings have the size of (18992, 300).

Regarding the embeddings for skill sets, which will be called skill embeddings, due to the large size (257450 phrases), a pipeline of PPA (vector operations using python *numpy*) + PCA (PCA function from python package $scikit-learn$) + PPA was done on the word embeddings mapping to each word in the skill phrases to reduce the size of the vectors to half size. The skill embeddings (in long shape, with dimensions of 257450 x 150) was created by taking average of the reduced vectors.

**Column selection**

Next, in order to conduct the column selection, the skill embeddings needed to transformed to wide shape. The process was done in two steps:

1. Divide the long skill embeddings into 3 subsets corresponding to 3 columns in the original dataset: $job\_description$, $job\_requirement$, and $job\_qualification$.

2. Merging 3 subsets into one matrix of skill embeddings in wide shape based on the unique id of *job_id* and *skill_id*.

Further, synthetic data were generated by running permutation per column then merged with the wide skill embeddings. $RandomForestClassifier$ from python package $scikit-learn$ was then applied to train on the merged matrix. The yielded feature importances are as follows:

```
There are 225/450 (~50.0%) most important features,
 with importance higher than 0.00219, including:
- 118 (~52.44%) features from job_description,
- 25 (~11.11%) features from job_requirement,
- 82 (~36.44%) features from job_qualification.
```

Thus, it was decided to omit the text embeddings representing *job_requirement* from the long embeddings. The final skill embeddings have the size of (189882, 150).

## Clustering and evaluation

### *Cluster tendency*

First, it was needed to conduct the Hopkins test, to see whether the dataset have the cluster tendercy or not. The Hopkins score ($H$) from python package pyclustertend was computed to measure the cluster tendency of both title embeddings and skill embeddings (Lachheb, 2021). According to the documentation of pyclustertend, if the Hopkins score is close to 0, it is useful to conduct clustering, and vice versa, if the Hopkins score is higher than 0.3, there seems to be no cluster tendency in the dataset.

Table 3 and Table 4 show the Hopkins statistics with different number of samples for testing. It can be concluded that both job titles and skill sets have cluster tendency; hence, clustering methods can be done afterward.

### *Clustering and Hyperparameters*

The clustering was done by different python packages, namely:

- k-means: $KMeans$ of $scikit-learn$ (Pedregosa et al., 2011).

Table 3

*Hopkins scores computed on title embeddings*

| dataset | dataset_size | n_samples | hopkins_score | test_duration |
|---|---|---|---|---|
| title_embeddings | (18992, 300) | 50 | 0.0616 | 0 days 00:00:03.524348 |
| title_embeddings | (18992, 300) | 100 | 0.0641 | 0 days 00:00:04.302260 |
| title_embeddings | (18992, 300) | 1000 | 0.0591 | 0 days 00:00:17.937140 |
| title_embeddings | (18992, 300) | 5000 | 0.0560 | 0 days 00:01:19.657116 |
| title_embeddings | (18992, 300) | 10000 | 0.0573 | 0 days 00:02:30.997003 |

Table 4

*Hopkins scores computed on skill embeddings*

| dataset | dataset_size | n_samples | hopkins_score | test_duration |
|---|---|---|---|---|
| skill_halfsize_embeddings | (189882, 150) | 50 | 0.0146 | 0 days 00:00:30.281267 |
| skill_halfsize_embeddings | (189882, 150) | 100 | 0.0219 | 0 days 00:00:33.222075 |
| skill_halfsize_embeddings | (189882, 150) | 1000 | 0.0223 | 0 days 00:01:31.979072 |
| skill_halfsize_embeddings | (189882, 150) | 5000 | 0.0244 | 0 days 00:06:07.380748 |
| skill_halfsize_embeddings | (189882, 150) | 10000 | 0.0241 | 0 days 00:11:27.120484 |

- Genie: *Genie* of *genieclust* (Gagolewski et al., 2016).
- HDBSCAN: $HDBSCAN$ of *hdbscan* (McInnes, Healy, & Astels, 2017).
- LDA: $LdaModel$ of *gensim* (Rehurek & Sojka, 2010).
- GSDMM: $MovieGroupProcess$ of $GSDMM$ (Yin & Wang, 2014).

For k-means and Genie, the only hyperparameter needed to tune is the number of clusters. A wide range from 2 to 200 were experimented to find out which was the optimal number of clusters.

For HDBSCAN, there is no need to pre-define the number of clusters, but

$min_samples$ and $min_cluster_size$ instead. The clustering with each hyperparameter ranging from 3 to 100 were iterated to figure out the best combination of hyperparameter.

For LDA and GSDMM, besides number of clusters, *alpha* and *beta*, ranging from 0 to 1, were needed to be tuned.

***Evaluation metrics***

The performance of clustering models were evaluated by intrinsic measures, namely: Silhouette score ($Silhouette\_score$), Calinski and Harabasz score ($CH\_score$), and Davies-Bouldin score ($DB\_score$). Silhouette score was the main metric to decide the best model. Formulas to compute each score are as following:

$$Silhouette\_score = \frac{a - b}{max(a, b)} \tag{3}$$

where $a$ is the mean distance to the points in the nearest cluster, and $b$ is the mean intra-cluster distance to all the points.

$$CH\_score = \left[\frac{\sum_{k=1}^{K} n_k \left\|c_k - c\right\|^2}{K - 1}\right] / \left[\frac{\sum_{k=1}^{K} \sum_{i=1}^{n_k} \left\|d_i - c_k\right\|^2}{N - K}\right] \tag{4}$$

where $K$ is number of clusters, $d_i$ is datapoint $i$, $n_k$ and $c_k$ are the number of points and centroid of the $k^{th}$ cluster respectively, $c$ is the global centroid, $N$ is the total number of data points.

$$DB\_score = \frac{1}{n}\sum_{i=1}^{n} max_{j \neq i}\left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right) \tag{5}$$

where $n$ is number of clusters, $\sigma_i$ is average distance of all points in cluster $i$ from the cluster centroid $c_i$, $d(c_i, c_j)$ is the distance between cluster centroid $c_i$ and $c_j$.

Besides, for topic modelling methods LDA and GSDMM, the average topic coherence score, given by Mazarura, de Waal, and de Villiers (2020), were also computed and compared between results of the two methods. The formula of coherence score for each topic, $T$, is as following:

$$coherence(T) = \sum_{(v_i, v_j) \in T} log \frac{D(v_i, v_j) + \epsilon}{D(v_j)} \quad (6)$$

where $v_i$ is word $i$ in topic $T$, $D(v_i, v_j)$ is the number of documents that $v_i$ and $v_j$ co-occur, $D(v_j)$ is the number of documents that $v_j$ occurs, $\epsilon$ is the smoothing parameter (which is set as 1 in the original paper).

## Results

## Discussion

## Conclusion

## Acknowledgements

References

Alelyani, S., Tang, J., & Liu, H. (2013). Feature selection for clustering: A Review. *Data clustering: algorithms and applications*, *29*(110-121), 144.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, *3*, 993–1022.

Boselli, R., Cesarini, M., Mercorio, F., & Mezzanzanica, M. (2018). Classifying online Job Advertisements through Machine Learning. *Future Generation Computer Systems*, *86*, 319–328. doi: 10.1016/j.future.2018.03.035

Breiman, L. (2002). Manual on setting up, using, and understanding random forests v3. 1. *Statistics Department University of California Berkeley, CA, USA*, *1*, 58.

Calanca, F., Sayfullina, L., Minkus, L., Wagner, C., & Malmi, E. (2019). Responsible team players wanted: An analysis of soft skill requirements in job advertisements. *EPJ Data Science*, *8*(1), 13. doi: 10.1140/epjds/s13688-019-0190-z

Campello, R. J., Moulavi, D., & Sander, J. (2013). Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining* (pp. 160–172). Springer.

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Nunes, C., & Jatowt, A. (2020, January). YAKE! Keyword extraction from single documents using multiple local features. *Information Sciences*, *509*, 257–289. doi: 10.1016/j.ins.2019.09.013

Dave, V. S., Zhang, B., Al Hasan, M., AlJadda, K., & Korayem, M. (2018). A Combined Representation Learning Approach for Better Job and Skill Recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (pp. 1997–2005). Torino Italy: ACM. doi: 10.1145/3269206.3272023

De Mauro, A., Greco, M., Grimaldi, M., & Ritala, P. (2018). Human resources for Big Data professions: A systematic classification of job roles and required skill sets. *Information Processing & Management*, *54*(5), 807–817. doi: 10.1016/j.ipm.2017.05.004

Djumalieva, J., Lima, A., & Sleeman, C. (2018). Classifying occupations according to

their skill requirements in job advertisements. *Economic Statistics Centre of Excellence Discussion Paper*, *4*, 2018.

Elghazel, H., & Aussem, A. (2015, January). Unsupervised feature selection with ensemble learning. *Machine Learning*, *98*(1), 157–180. doi: 10.1007/s10994-013-5337-8

Florescu, C., & Caragea, C. (2017). Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1105–1115).

Gagolewski, M., Bartoszuk, M., & Cena, A. (2016, October). Genie: A new, fast, and outlier-resistant hierarchical clustering algorithm. *Information Sciences*, *363*, 8–23. doi: 10.1016/j.ins.2016.05.003

Grira, N., Crucianu, M., & Boujemaa, N. (2005). Unsupervised and Semi-supervised Clustering: A Brief Survey. In *In 'A Review of Machine Learning Techniques for Processing Multimedia Content', Report of the MUSCLE European Network of Excellence (FP6.*

Han, J., & Kamber, M. (2012). *Data mining: Concepts and techniques* (3rd ed ed.). Burlington, MA: Elsevier.

Lachheb, I. (2021, February). *Lachhebo/pyclustertend: V1.5.0.* Zenodo. doi: 10.5281/ZENODO.4566734

Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on knowledge and data engineering*, *17*(4), 491–502.

Liu, L., Kang, J., Yu, J., & Wang, Z. (2005, October). A comparative study on unsupervised feature selection methods for text clustering. In *2005 International Conference on Natural Language Processing and Knowledge Engineering* (pp. 597–601). doi: 10.1109/NLPKE.2005.1598807

Liu, T., Liu, S., Chen, Z., & Ma, W.-Y. (2003). An Evaluation on Feature Selection for Text Clustering. , 8.

Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015, June). Recommender system application developments: A survey. *Decision Support Systems*, *74*, 12–32. doi: 10.1016/j.dss.2015.03.008

Mazarura, J., de Waal, A., & de Villiers, P. (2020, April). A Gamma-Poisson Mixture Topic Model for Short Text. *Mathematical Problems in Engineering*, *2020*, e4728095. doi: 10.1155/2020/4728095

McInnes, L., Healy, J., & Astels, S. (2017, March). Hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, *2*(11), 205. doi: 10.21105/joss.00205

Mhamdi, D., Moulouki, R., El Ghoumari, M., Azzouazi, M., & Moussaid, L. (2020). Job Recommendation based on Job Profile Clustering and Job Seeker Behavior. *Procedia Computer Science*, *175*, 695–699. doi: 10.1016/j.procs.2020.07.102

Mihalcea, R., & Tarau, P. (2004). Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing* (pp. 404–411).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*(85), 2825–2830.

Poch, M., Bel Rafecas, N., Espeja, S., & Navio, F. (2014). Ranking job offers for candidates: Learning hidden knowledge from big data. In *In: Calzolari N, Choukri K, Declerck T, Loftsson H, Maegaard B, Mariani J, Moreno A, Odijk J, Piperidis S, editors. Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014); 2014 May 26-31; Reykjavik, Iceland. Paris: European Language Resources Association; 2014. p. 2076-82.* ACL (Association for Computational Linguistics).

Qiang, J., Qian, Z., Li, Y., Yuan, Y., & Wu, X. (2020). Short Text Topic Modeling Techniques, Applications, and Performance: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 1–1. doi: 10.1109/TKDE.2020.2992485

Qin, Y., Ding, S., Wang, L., & Wang, Y. (2019, October). Research Progress on

Semi-Supervised Clustering. *Cognitive Computation*, *11*(5), 599–612. doi: 10.1007/s12559-019-09664-w

Raunak, V., Gupta, V., & Metze, F. (2019, August). Effective Dimensionality Reduction for Word Embeddings. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)* (pp. 235–243). Florence, Italy: Association for Computational Linguistics. doi: 10.18653/v1/W19-4328

Reddy, Y. C. A. P., Viswanath, P., & Reddy, B. E. (2018, February). Semi-supervised learning: A brief review. *International Journal of Engineering & Technology*, *7*(1.8), 81–85. doi: 10.14419/ijet.v7i1.8.9977

Rehurek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. In *In Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks.* Citeseer.

Rose, S., Engel, D., Cramer, N., & Cowley, W. (2010). Automatic keyword extraction from individual documents. *Text mining: applications and theory*, *1*, 1–20.

Smith, A. (2015). Searching for work in the digital era. *Pew Research Center*, *19*.

Sun, X. (2014, August). Textual Document Clustering Using Topic Models. In *2014 10th International Conference on Semantics, Knowledge and Grids* (pp. 1–4). Beijing, China: IEEE. doi: 10.1109/SKG.2014.27

Suyal, H., Panwar, A., & Singh Negi, A. (2014, June). Text Clustering Algorithms: A Review. *International Journal of Computer Applications*, *96*(24), 36–40. doi: 10.5120/16946-7075

Vinel, M., Ryazanov, I., Botov, D., & Nikolaev, I. (2019). Experimental comparison of unsupervised approaches in the task of separating specializations within professions in job vacancies. In *Conference on Artificial Intelligence and Natural Language* (pp. 99–112). Springer.

World Economic Forum. (2020). The Future of Jobs Report 2020. World Economic Forum, Geneva, Switzerland.

Yin, J., & Wang, J. (2014, August). A dirichlet multinomial mixture model-based approach for short text clustering. In *Proceedings of the 20th ACM SIGKDD*

*international conference on Knowledge discovery and data mining* (pp. 233–242). New York New York USA: ACM. doi: 10.1145/2623330.2623715