Content Based Job Recommendation System Using Word Embedding Shallow Neural Network, TF-IDF & KNN

Md Tanzir Altaf, Linkstaff Co.,Ltd,Tokyo, Japan.
Arafat Ullah, Linkstaff Co.,Ltd ,Dhaka, Bangladesh.
Nabapadma Sarker, Linkstaff Co.,Ltd,Tokyo, Japan.

ABSTRACT

Business sectors are vastly improved because of online marketing and addition of machine learning boost up this improvement further which is recommending products to the users based on their explicitly or implicitly provided rating and this only works for those who have previously interacted with the products, through recommendation system useful products could be suggested to new users' by only gathering their personal information. The purpose of this study is to build a recommendation engine for doctor's living in Japan for reducing their hardship and expenses and saving their time in searching for suitable jobs. To serve this purpose several approaches were explored and a comparative study also performed among them in this research. Seven various models were introduced and it includes Average Word2vec, TF-IDF algorithms' sequential executions with cosine similarity and KNN individually also one sequential model of Average Word2vec, TF-IDF and cosine similarity combined and same sequential model with KNN and the remaining model was an ensemble method of three distinct cosine similarity values obtained from the three sequential models with cosine similarity mentioned earlier. After that all the test results were compared with actual data through a performance measuring algorithm introduced in this study. Based on this measurement a comparative study was performed among those models. It helps in identifying suitable and efficient models and also draws some limitations which would be the key element for further analysis.

KEYWORDS

Recommendation, Average Word2Vec Model, Mecab Tokenizer and POS tagging, TF-IDF, KNN, Cosine Similarity

INTRODUCTION

Customer satisfaction is one of the key factors in rapid business development. What buyer wants to buy what he like more what he needed if all of those could recommend beforehand and products could be collected accordingly then it's optimistic in 100% revenue. This feat could achieve in two ways such as ask for rating the products to the users' explicitly or extracting their likeliness implicitly by analyzing their previous interaction with the products. In explicit ratings there is a huge drawback since many customers rate products randomly which doesn't reveal their likeness. In implicit ratings which particular products a specific user frequently bought previously by analyzing this, those particular products could be recommended to the specific user. This feat could also be achieved through traditional machine learning or data mining but these techniques are not feasible for recommending products to the newcomer. A Recommendation system is the appropriate medium for achieving this feat. In the recommendation system when a new user registered the site only receiving his personal details which were much required for being registered which types of products he may like more or may buy more this could be extracted through several recommendation techniques which is why it's more beneficial than traditional machine learning or data mining techniques. Also, for this reason the recommendation system's use is increasing day by day in every tiny, big business sector which were previously limited only to mega tech giants like Google, Netflix etc. The majority of such systems are applied in the field of e-commerce for e.g., product recommendations [1]. Many companies are now using this system for recruiting new candidates online. Japan has several such types of companies who use a job recommendation system. But in this study, we only focus on the doctor's job recommendation in Japanese market. Many medical students after completing their MBBS degree search for jobs which will be well suited with their portfolio. Also, many doctors' want to switch their job and for this they have to search for a new job which matches their profile. For both the medical students and the doctors searching for those jobs manually and attending interviews are very time consuming, expensive and also troublesome. For medical students sometimes it's hard to find those jobs which will exactly match their profile. But if there is an online system which could filter only those jobs which are appropriate based on their profile will help to reduce this trouble. And the purpose of this study is to serve them this feature. Recommendation systems are of two types; content-based filtering and collaborative filtering. And in this study content-based filtering was used. Content-based filtering selects the right information for users by comparing representations of searching information to representations of contents of user profiles which express interests of users. Content-based information filtering has proven to be effective in locating textual items relevant to a topic using techniques, such as Boolean queries. In this context the main challenge is extracting the relevant feature from Japanese context in our collected data which has lots of attributes containing null values. To carry out this work, Firstly, collected our 32000 Japanese text dataset is consider only 'nouns'(e.g. クリニック(Kurinikku) "clinic" and 手術 (Shujutsu)"surgery") — non-conjugating; no marking for number (e.g. singular vs. plural)[20] or grammatical gender or definiteness ; highly productive right headed noun compounding via simple concatenation (e.g. 機械 (kikai) "machine" + 翻訳 (Hon'yaku) "translation" +協会 (kyōkai) "association" =機械翻訳協会 (kikai hon'yaku kyōkai) "machine translation association") or linking with the no case marker (e.g. 会社(kaisha) "company" + 人(hito) "person" = 会社の 人(kaisha no hito) "company person")[2]. Currently, there are many libraries to tokenize the Japanese text like, juman, Juman++, Mecab, Konoha. After tokenization, we use word2vec skip-gram word embedding model and create a 100-vector dimension of each word. Each job description is a sentence or sequence of words. Total seven different models were introduced in this study. Three different vectorization algorithms which were average word2vec, TF-IDF vectorization and another approach was combination of these two methods which was TF-IDF*word2vec, were used to convert the words into vector dimension. Then to search the relevant information of word vectors KNN classifier and cosine similarity were sequentially used with the previous three algorithms. And in the last model, an ensemble technique was introduced by averaging three models output values. Then all seven models were tested using only a single feature of doctors' data, then a new performance measuring algorithm was introduced in this study where all those test results were taken into account and then compared with actual job data using cosine similarity. This performance measure provides two outcomes; one is how suitable those models work and the other one is a comparison among those models.

This article is divided into six sections to better clarify the overall idea. Next section broadly introduces previous contributions and ideas which have been explored by many academics as well as independent researchers. The used data collection mechanism, resources, feature selection, are described in section 3. Section 4 refers to the proposed approaches that were used in this work to deal with the problem under study. The experimentation and results are presented and discussed in section 5. Lastly, section 6 presents the overall conclusions and the future direction of the research in the area of the job recommendation system.

LITERATURE REVIEW

In the last 16 years, more than 200 research articles were published about *research-paper recommender systems*. We found that more than half of the recommendation approaches applied content-based filtering (55%). Collaborative filtering was applied by only 18% of the reviewed approaches, and graph-based recommendations by 16%. Other recommendation concepts included stereotyping, item-centric recommendations, and hybrid recommendations [3]. Content-based systems are recommendation systems that are based on the features of the item we're trying to recommend. When talking about jobs, this includes for example the subject of the job or how elaborative the job description is, so in terms of finding the frequently appearing similarity contexts and this is where word2vec comes in. Among content-based filtering there is much research conducting word embedding methods depending on several hyper-parameters that have a crucial impact on the quality of embeddings. For this reason, Mikolov et al. [4, 5] and Pennington et al. [6] –the inventors of the popular low-dimensional embedding word2vec and GloVe, respectively – have deeply studied the optimization of the embedding parameters, mainly the vector dimension and the context size [7]. In this research, we face some challenges in the process of Japanese morphological analysis. Japanese is known for having three distinct orthographies: kanji, hiragana, and katakana. The latter two are occasionally grouped into the same category, kana. Besides Japanese context written without space between the words, due to the fact that Japanese word extraction is a little bit challenging. When conducting Japanese tokenization for business applications, in the majority of cases MeCab [9] or Kuromoji[10] (There-implementation of MeCab) are used. MeCab can process text at excellent speed, however, its functions are limited to segmentation, POS tagging, and lemmatization; Users need to pre-/post-process the text by themselves. Currently, there are several tools available (e.g., juman, juman++, fugashi, sudachi etc.) online to handle this kind of problem. Representation of words as continuous vectors has a long history. A very popular model architecture for estimating neural network language model (NNLM) was proposed in [11], where a feedforward neural network with a linear projection layer and a non-linear hidden layer was used to learn jointly the word vector representation and a statistical language model. Another interesting architecture of NNLM was presented in [12, 13], where the word vectors are first learned using a neural network with a single hidden layer. The word vectors are then used to train the NNLM. Thus, the word vectors are learned even without constructing the full NNLM. In paper [14], authors proposed two new models e.g., CBOW and Skip-gram architectures for learning distributed representations of words that try to minimize computational complexity. Word2vec is a simple neural network model with a single hidden layer. It predicts the adjacent words for each and every word in the sentence or corpus. We need to get the weights that are learned by the hidden layer of the model and the same can be used as word embedding. Word representing a vector is a general predictive model for learning vector representations of words [14]. These vector representations, also called word embedding, capture distributional semantics and co-occurrence statistics [4]. In this research paper, we use a skip-gram model. More precisely, we use each current word as an input to a log-linear classifier with a continuous projection layer, and predict words within a certain range before and after the current word. In this research paper [15], uses TF-IDF Content-based recommender System. This filtering process uses the features weighting method known as Term Frequency-Inverse Document Frequency (TF-IDF) for its content-based recommender system. This method filters the data set for items which have keyword(s) in their descriptions which match the user's profile indicators. Therefore, for dealing with the exponential growth of jobs in the Japanese market, we specifically focus on developing a new hybrid ensemble method approach for efficient analysis as well as finding the relevant features of the job to analyze the data from the doctor's profile.

ADAPTED METHODS

In order to accomplish this research, two methods were used; Word2Vec skip-gram model for D-Dimension vector representation of word and TF-IDF for single dimension vector representation of word. Besides we use another third approach which is multiple of these two methods e.g., TF-IDF*Word2Vec. For similarity measurement in word vector space, we use Cosine Similarity and KNN Classifier. Methods description are given below:

Word2vec Skip-gram Model

Word2Vec is not a singular algorithm, rather, it is a family of model architectures and optimizations that can be used to learn word embeddings from large text datasets. Embeddings learned through Word2Vec have proven to be successful on a variety of downstream natural language processing tasks. Rather than predicting a word based on its context, Skip-gram aims to predict the context based on one word [4].

Word2Vec is a simple shallow neural network model with a single hidden layer. It predicts the adjacent words for each and every word in the sentence or corpus. The work in [14] is a popular choice for pre-training the projection matrix W $\in$ $R^{d\times|V|}$ where d is the embedding dimension with the vocabulary V. As an unsupervised task that is trained on raw text, it builds word embeddings by maximizing the likelihood that words are predicted from their context or vice versa. In our approach, we use a skip-gram model. The skip-gram model's objective function is to maximize the likelihood of the prediction of contextual words given the center word. More formally, given a document of T words, we wish to maximize

$$L = \frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-c \le j \le c, \\ j \ne 0}} \log p\,(w_{t+j} \mid w_t) \;\ldots\ldots\ldots\ldots\ldots\ldots \; (1)$$

Where c is a hyperparameter defining the window of context words. To obtain the output probability $p(w_o|w_i)$, the model estimates a matrix O $\in R^{|V|\times dw}$, which maps the embeddings r_{wi} into a |V |-dimensional vector o_{wi}. Then, the probability of predicting the word w_o given the word w_i is defined as:

$$p\,(w_o \mid w_i) = \frac{e^{o_{w_i}(wo)}}{S_{w\in V} e^{o_{w_i}(w)}} \;\ldots\ldots\ldots\ldots\ldots\ldots \; (2)$$

This is referred to as the SoftMax objective. However, for larger vocabularies it is inefficient to compute o_{wi}, since this requires the computation of a |V | × d_w matrix multiplication. Solutions for problems are addressed in the Word2Vec by using the hierarchical SoftMax objective function or resorting to negative sampling [16]. The skip-gram model uses a single output matrix O $\in$ $R^{|V|\times d}$ to predict every contextual word w−c, ..., w−1, w1, ..., wc, given the embeddings of the center word w0. Our approach adapts the model so that it is sensitive to the positioning of the words.

In simpler terms, Word2Vec takes the word and returns a vector in D-dimensional space.

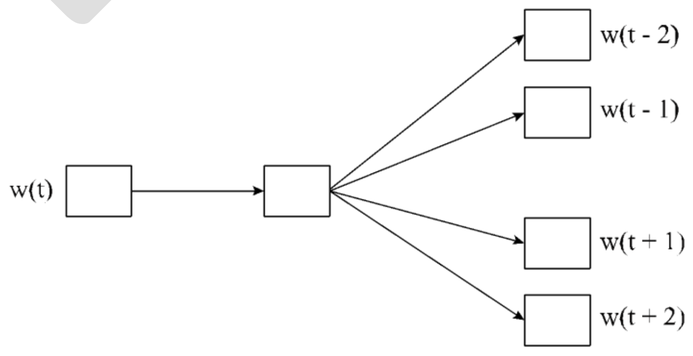

Figure 1: Skip-gram Model

Please note, Word2Vec provides the word embeddings in low dimensionality (50–500) which are dense (it's not a sparse matrix, most values are non-zero).

To train our corpus data we use the gensim library word2vec skip-gram module. This module implements the word2vec family of algorithms, using highly optimized C routines, data streaming and Pythonic interfaces.[17]

We used 100-dimension vectors for this recommendation engine. As we mentioned above, Word2Vec is good at capturing semantic meaning and relationships [18]. Training our own word embeddings also requires a large dataset. As I've a large dataset with 32000 rows and 118 columns so conducting this research authors trained the model for word embeddings with 100 epoch and given parameters was: vector_size=100 (Dimensionality of the word vectors), minicoat=2 (ignores all words with total frequency lower than this), window=3 (Maximum distance between the current and predicted word within a sentence), sample=1e-3 (the threshold for configuring which higher-frequency words are randomly down sampled), alpha=0.03 (The initial learning rate), min_alpha=0.007 (Learning rate will linearly drop to min_alpha as training progresses.), negative=20 (specifies how many "noise words" should be drawn), sg ({0, 1}, optional) =1 (Training algorithm: 1 for skip-gram; otherwise CBOW).

After initializing the model and training for 100 epochs, word embeddings helped to get the vectors for the words we need. It is a large collection of key-value pairs, where keys are the words in the vocabulary and values are their corresponding word vectors. For data exploration we use the sklearn TSNE model. In this model we extract few Japanese word ['眼科', '内科', '整形外科','脳神経','内分泌'] and draw their corresponding vector space word map in 2D representation.

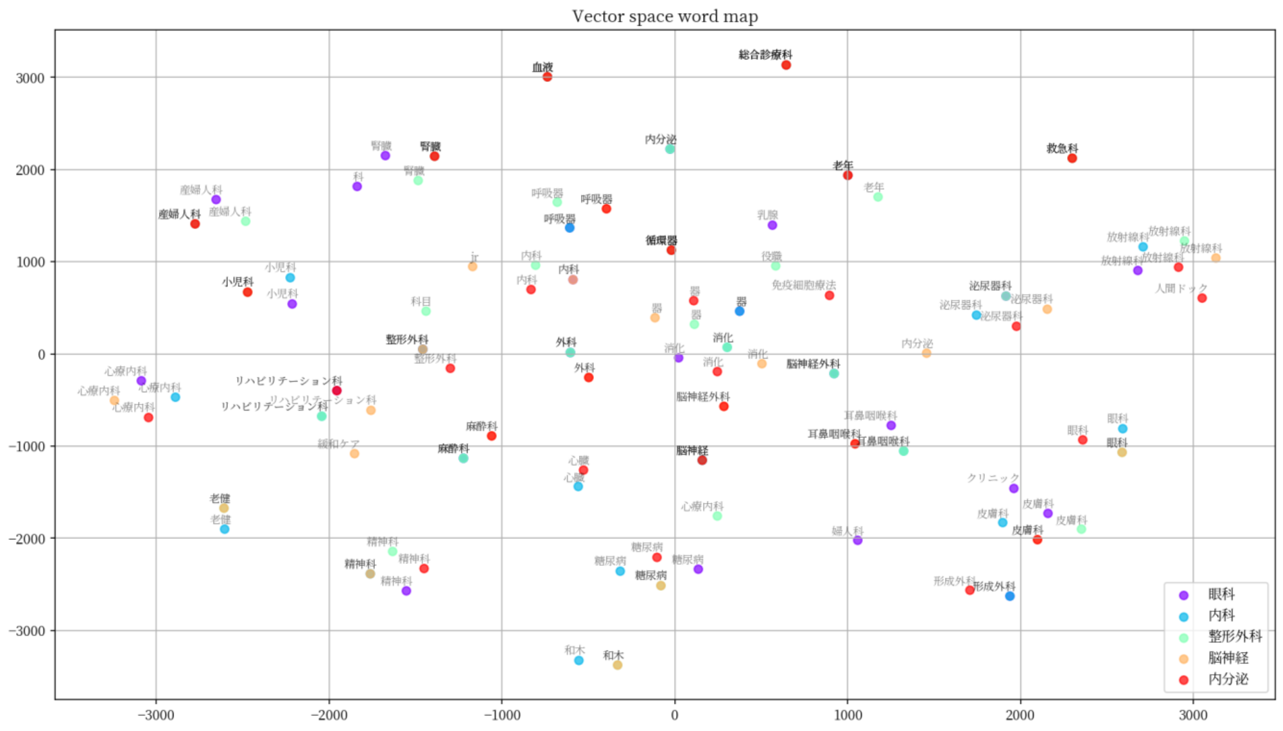

Figure 2: Vector space word map in 2D

For visualization of the trained word vector, we use UMAP. Uniform Manifold Approximation and Projection (UMAP) is a dimension reduction technique that can be used for visualization similarly to t-SNE, but also for general nonlinear dimension reduction. The algorithm is founded on three assumptions about the data

i. The data is uniformly distributed on Riemannian manifold;
ii. The Riemannian metric is locally constant (or can be approximated as such);
iii. The manifold is locally connected.

From these assumptions it is possible to model the manifold with a fuzzy topological structure. The embedding is found by searching for a low dimensional projection of the data that has the closest possible equivalent fuzzy topological structure [19].

We use UMAP library and set a parameter to n_neighbors = 30 (This parameter controls how UMAP balances local versus global structure in the data), min_dist = 0.0 (The parameter controls how tightly UMAP is allowed to pack points together), n_components = 2 (This parameter option that allows the user to determine the dimensionality of the reduced dimension space we will be embedding the data into), random_state=42 (If int, random_state is the seed used by the random number generator).

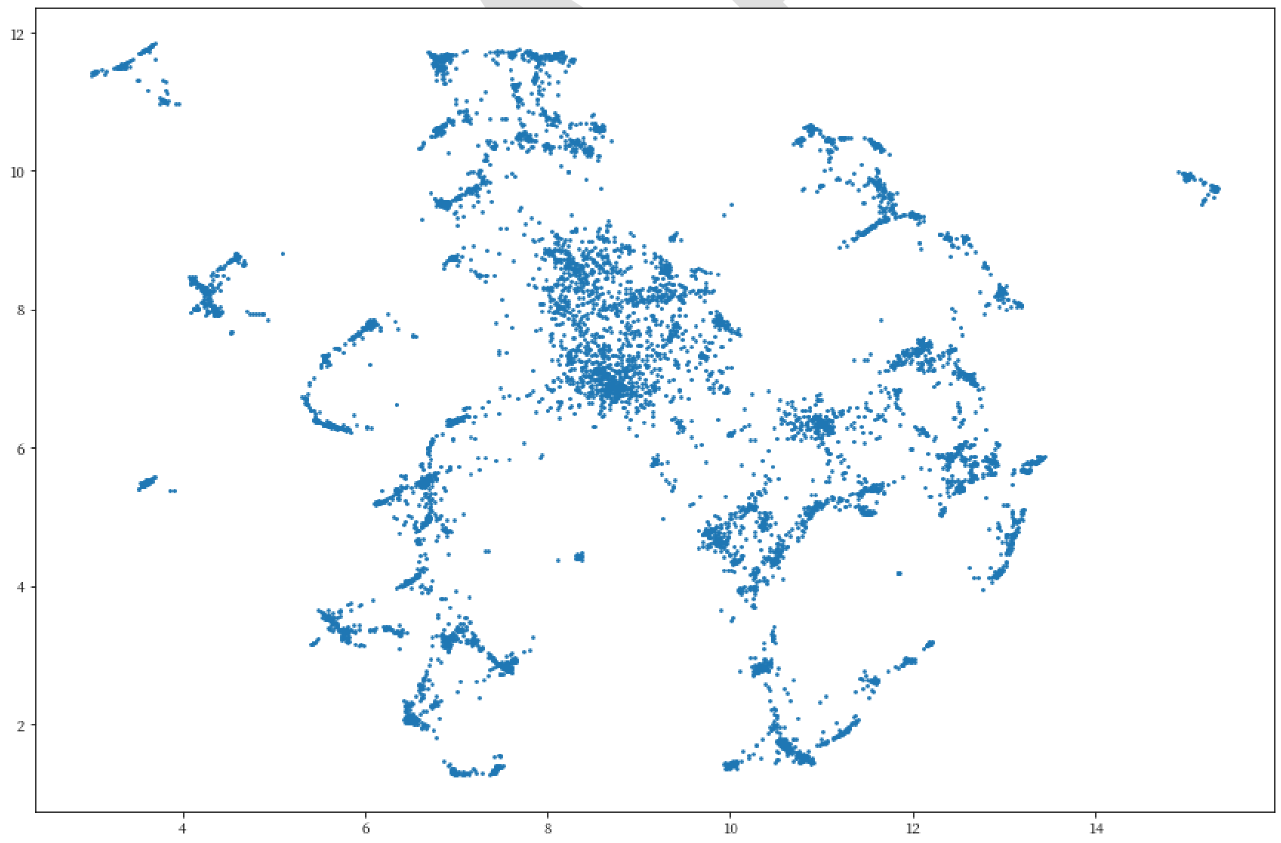

Figure 3: Graph plot from 100D to 2D vector space clustering

Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is a term frequency-inverse document frequency information retrieval algorithm. This method has been successful in topic identification over large text corpus, but its performance decreases when applied over small ones as those commonly found in job descriptions. However, TF-IDF has been applied to deal with recommendation obtaining interesting results in paper [20], [21]. TF-IDF assigns weights to the words as a statistical measure used to evaluate the relevance of a word in a document of a corpus. This relevance is proportional to the number of times a word appears in the document and inversely proportional to the frequency of the word in the corpus. It helps to calculate the importance of a given word relative to other words in the document and in the corpus. But the basic equation of this algorithms is written below:

$$TFIDF(word, doc) = TF(word, doc) * IDF(word) \quad \dots\dots\dots\dots\dots\dots (3)$$

Therefore, in this method, two matrices have to be calculated, one containing the inverse document frequency of a word in the whole corpus of documents and another containing the term frequency of each word in each document. The formula to calculate both of them are as follows:

$$TF\ (word, doc) = \frac{Frequency\ of\ word\ \in\ the\ doc}{No.\ of\ words\ \in\ the\ doc} \quad \dots\dots\dots\dots\dots\dots (4)$$

$$IDF\ (word) = log_e\left(1 + \frac{No.\ of\ docs}{No.\ of\ docs\ with\ word}\right) \dots\dots\dots\dots\dots\dots (5)$$

In our research, we use sklearn Tfidf Vectorizer model with parameter: min_df = 1 (When building the vocabulary, ignore terms that have a document frequency strictly lower than the given threshold.), lowercase=False (Convert all characters to lowercase before tokenizing, if value is true otherwise will not convert), stop_words = jp_stopwords (If a list, that list is assumed to contain stop words, all of which will be removed from the resulting tokens).

After training the list of documents and the TF-IDF score for each unique word in the entire corpus. Then plot TF-IDF scores for a dataset of texts in 2D. For clustering data point we use K-means and visualize the data point of TF-IDF vectors we use PCA from scikit-learn.

K-means is an algorithm designed to find coherent groups of data, a.k.a. clusters. In a general sense, k-means clustering works by assigning data points to a cluster centroid, and then moving those cluster centroids to better fit the clusters themselves. To run an iteration of k-means on our dataset, we first randomly initialize k (=10) number of points to serve as cluster centroids.

On the other hand, PCA performs dimension reduction by discarding the PCA features with lower variance, which it assumes to be noise, and retaining the higher variance PCA features, which it assumes to be informative.

To use PCA for dimension reduction, we need to specify how many PCA features to keep. In this experiment, we specify n_components=2 when creating a PCA model tells it to keep only the first two PCA features. A good choice is the intrinsic dimension of the dataset, here visual representation data point of TF-IDF vectors:

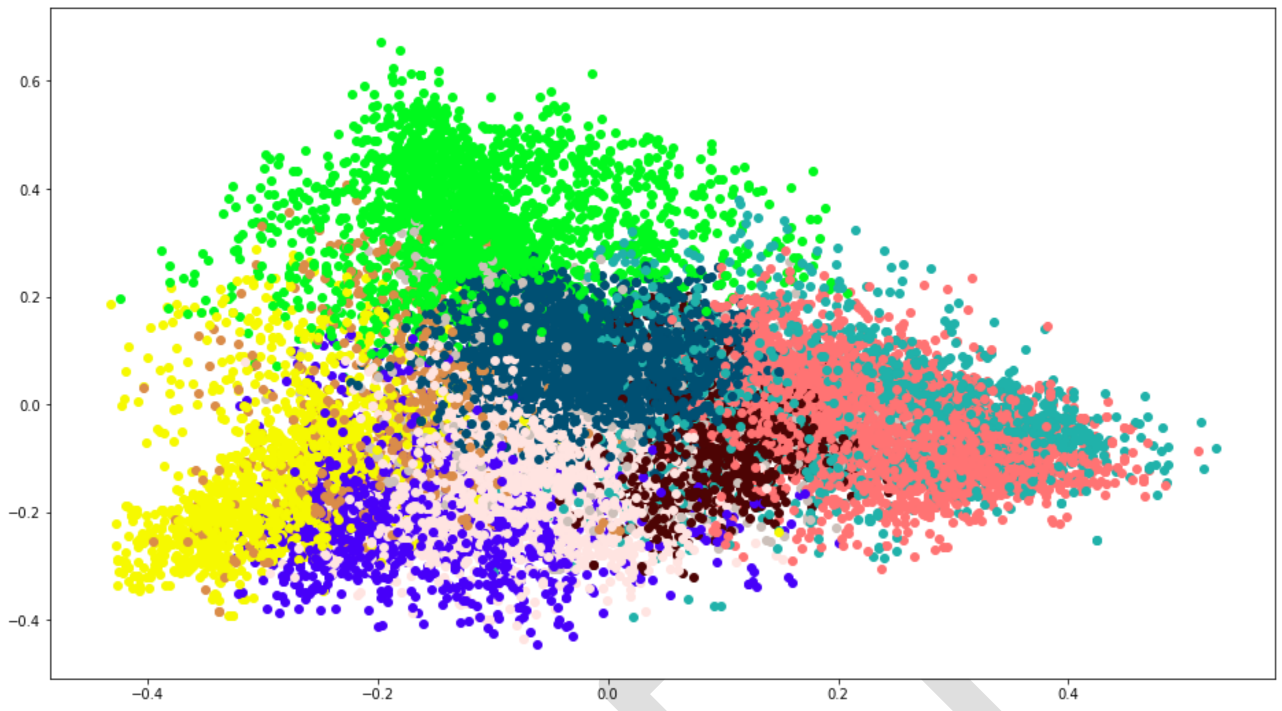

Figure 4: Data point of TF-IDF vector

Cosine Similarity:

In our problem domain, we need to convert the job descriptions into a vector and find the cosine similarity using the following equation.

$$similarity\ (a, b) = \cos\ (a, b) = \frac{a\ .\ b}{||a|| \times ||b||} \ldots\ldots\ldots\ldots\ldots\ (6)$$

Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis. Calculate the similarity between these trained word vectors to recommend jobs. Each job description is a sentence or sequence of words.

KNN classifier:

In statistics, the k-nearest neighbors algorithm (KNN) is a non-parametric classification method. It is used for classification and regression. In both cases, the input consists of the k closest training examples in the data set. The output depends on whether KNN is used for classification or regression.

In KNN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

KNN is a type of classification where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then normalizing the training data can improve its accuracy dramatically.

Just like almost everything else, KNN works because of the deeply rooted mathematical theories it uses. When implementing KNN, the first step is to transform data points into feature vectors, or their mathematical value. The algorithm then works by finding the distance between the mathematical values of these points. The most common way to find this distance is the Euclidean distance between two points in the plane with coordinates d (p, q) and d (q, p) can be calculated [22]: as shown below.

$$d(p, q) = d(q, p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \cdot \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (7)$$

KNN runs this formula to compute the distance between each data point and the test data. It then finds the probability of these points being similar to the test data and classifies it based on which points share the highest probabilities. In this experiment we use sklearn KNN model where we set the parameter n_neighbors=5 (find the first 5 nearest object), metric='euclidean' (measure the distance using euclidean distance) which takes the word2vec average word embedding vector of interest, the list with all embedding, and the number of similar jobs to retrieve. And the model outputs the dictionary IDs and distances.

PROPOSED METHODOLOGY

The main aspect of using this terminology in this experiment is to represent words into vector space and calculate the similarity of query data from those trained word vectors. In this research, we have tried 3 text to vector representation models and two similarity methods to find the most similar jobs with a doctor's profile. There are 2 phases on our methodology train and prediction. The overall framework of the adopted approach is illustrated in figure 5.

Training Phase:

We made a Combination of job and doctor data for training the model.

Data preprocessing phase:

We collected all the data from a website named e-doctors. From the vast number of features of jobs, we selected the most related fields with doctors and those have a good pattern over time. We combined the jobs and doctor’s data using union operation to remove duplicates, make a common set of features

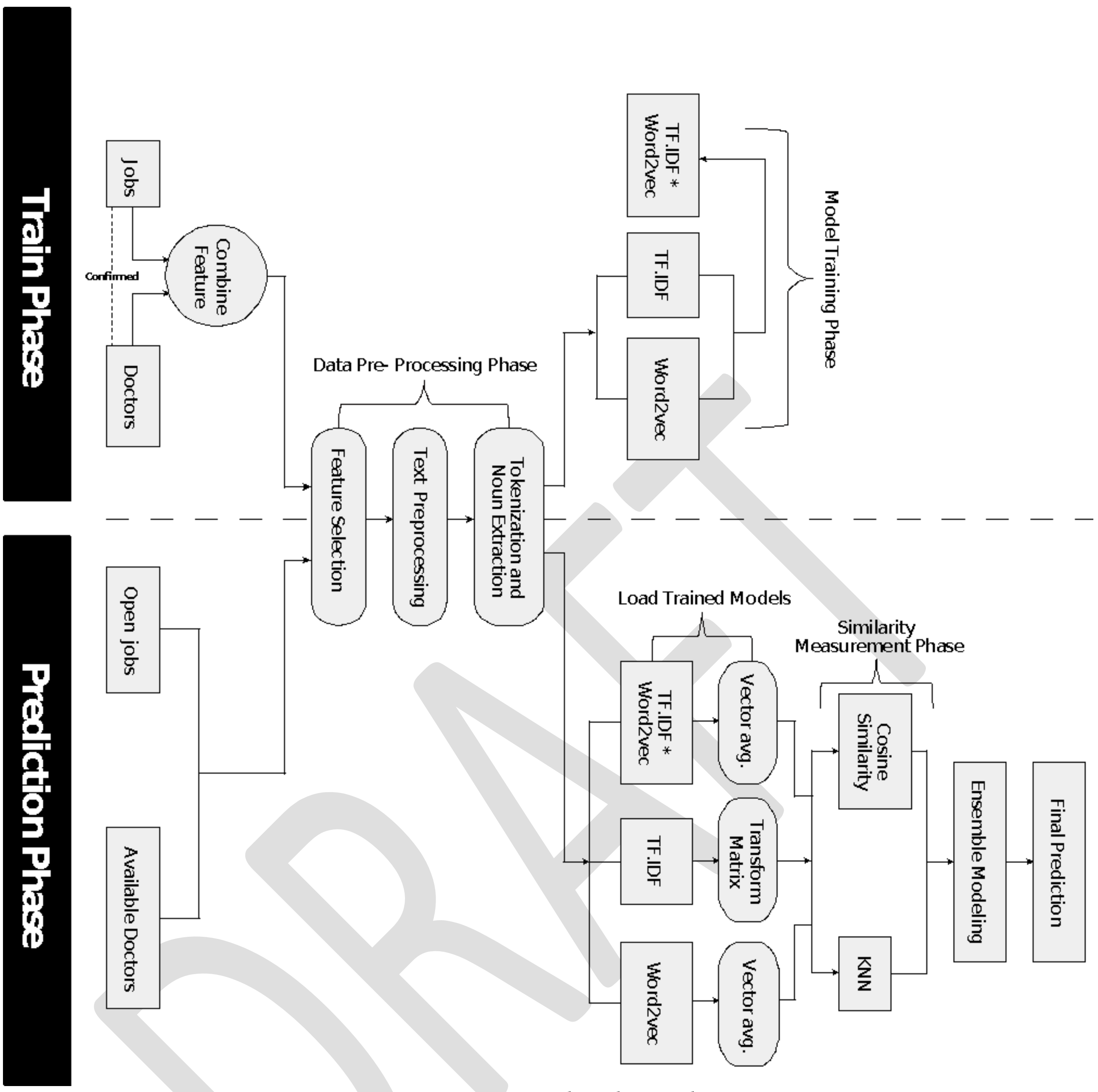

Figure 5: Train and Prediction phase

and concept the features to train data corpus (see figure 9). We used a few pre-processing techniques: null and empty row removal, Normalizing, punctuation removal and finally extracting nouns from remaining text (Here we use mecab-ipadic-neologd for POS tagging). For normalization we applied 'NFKC'. Then made a train corpus for models. After completing the data pre-processing, we got a structure data which we have visualized by using matplotlib to check the word frequency in accordance with the column. Some frequency distribution graphs are given below.

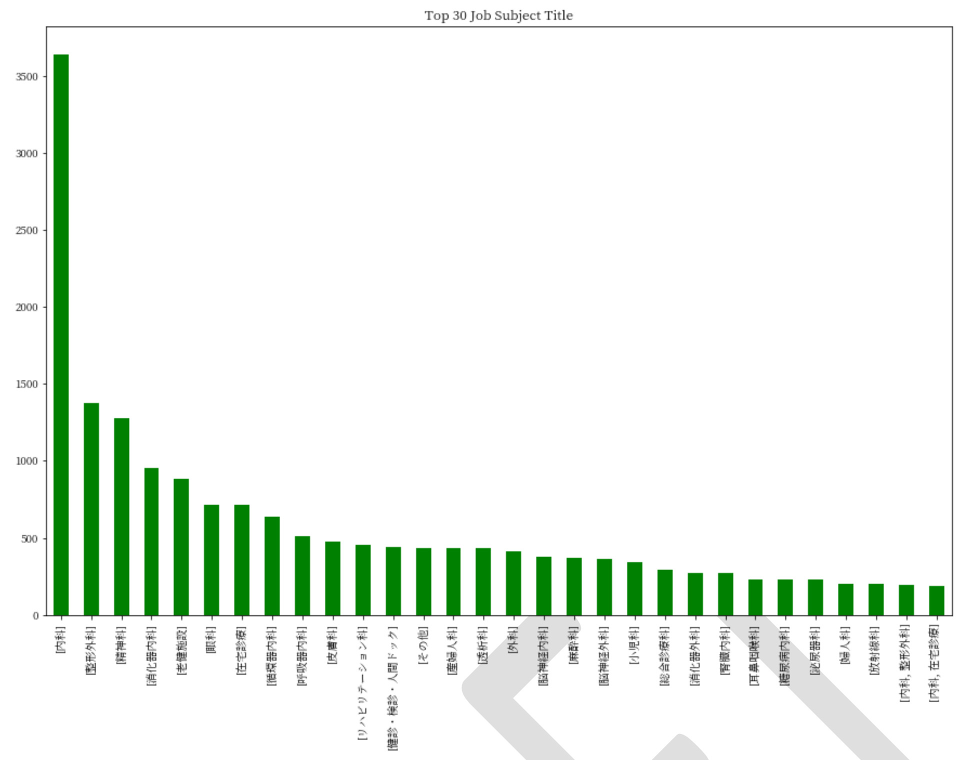

Figure 6: Top 10 job subject Title

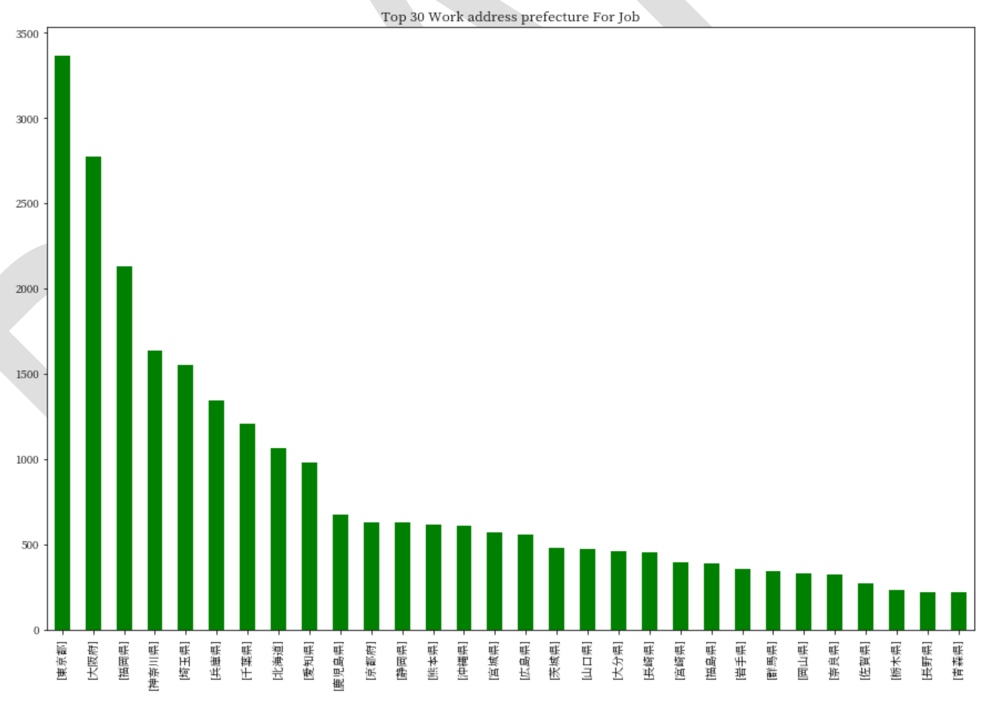

Figure 7: Top 10 prefecture for jobs

Then we combine our feature columns into one column which diagram is given below in figure 8.

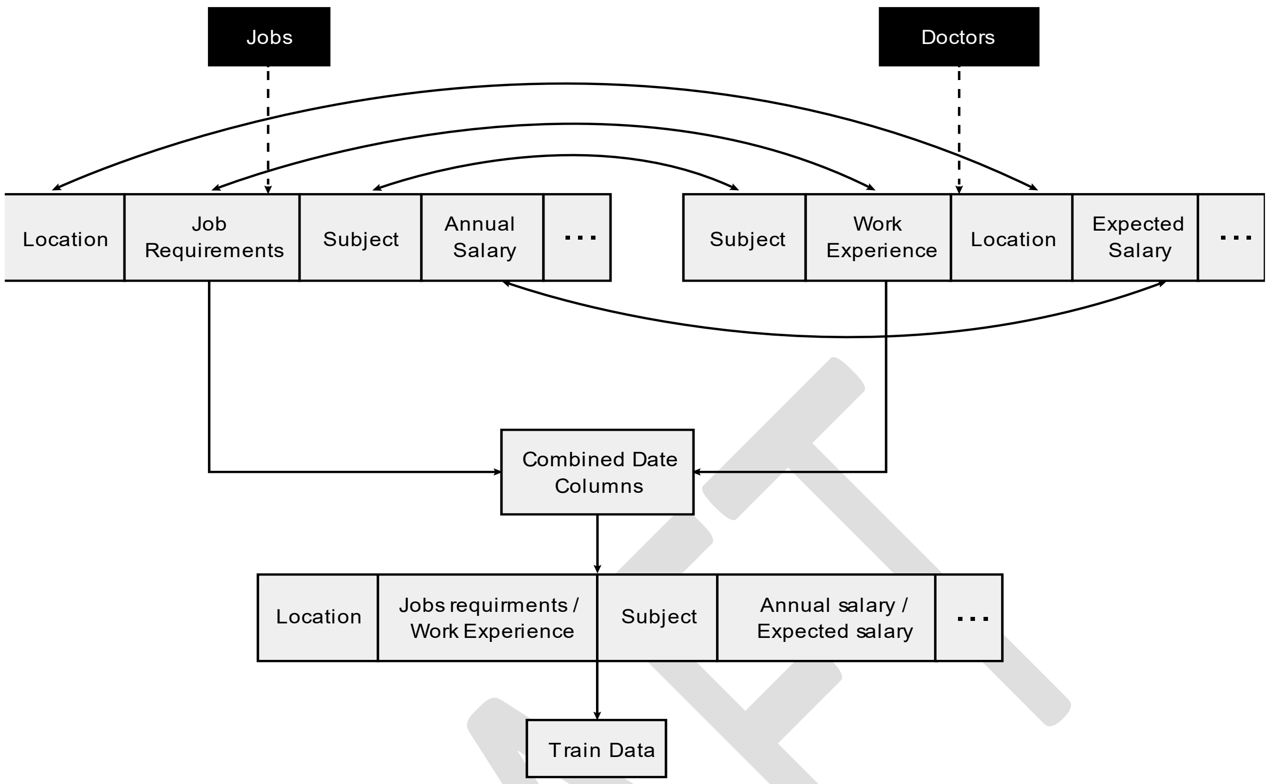

Figure 8: Features Combination

J_{loc} = job location

J_{req} = job requirements

J_{sub} = job subject

$J_{annual\ salary}$ = job Annual salary

D_{loc} = doctor location

$D_{work\ exp.}$ = Doctor work experience

D_{sub} = doctor subject

$D_{expec.\ salary}$ = doctor Expected Salary

Combining columns:

$C_{loc} = j_{loc} \cup D_{loc}$

$C_{work\ req./exp.} = j_{req} \cup D_{work\ exp.}$

$C_{sub} = j_{sub} \cup D_{sub.}$

$C_{annual\ salary} = J_{annual\ salary} \cup D_{expec.\ salary}$

$C_{training\ data\ corpus} = C_{loc} + C_{work\ req./exp.} + C_{sub} + C_{annual\ salary}$

To visualize the corpus data, we use word cloud. A word cloud (also known as a tag cloud) is a visual representation of words. Cloud creators are used to highlight popular words and phrases based on frequency and relevance. They provide you with quick and simple visual insights that can lead to more in-depth analyses.

内科
給与
交渉
外科
退職
求人
病院
カルテ
医療
診療
クリニック
地域
消化
インセンティブ
jr
オンコール
リハビリテーション
民間
手当て
役職
東京
勤務
年齢
整形
管理
呼吸
神経
特選
大阪
循環
透析
神奈川
検診
在宅
電子
病棟
千葉
救命
住宅
兵庫
無し

Figure 9: word2vec data visualization in word cloud

Prediction Phase:

In this phase, predictions are two types, for open jobs, how many doctors fit on those jobs then first 5 doctors have been picked as best fit. Another type is for available doctors how many open jobs fit and the first 5 jobs have been picked as best fit. For example, 10000 open jobs documents and 1 specific doctor document, we want to find the most similar jobs that fit most with that doctor. It can be visualized in figure8. First data preprocessing then load all models and make document transformation using **TF-IDF** and average vectors for the documents using **word2vec** and TF-**IDF * Word2Vec** model, then cosine similarity and KNN algorithm have been applied for similarity checking and at last average ensemble method has been applied for final prediction.

A. Word2Vec Average

Word2vec takes the word and gives a D-dimension vector. First, we need to split the sentences into words and find the vector representation for each word in the sentence. As we know, Word2vec takes the word and gives a D-dimension vector. First, we need to split the sentences into words and find the vector representation for each word in the sentence. For example, we have a sentence which consists of 23 words. Let's denote the words as w1, w2, w3, w4 …w23. Let's Calculate Word2Vec for all the 23 words. Then, sum all the vectors and divide the same by a total number of words in the description (n). It can be denoted as v1 and calculated as follows:

$$\frac{W2V(w_1) + W2(w_2) + ...+ W2V(w_{23})}{N} \quad \ldots\ldots\ldots\ldots\ldots\ldots \; (8)$$

Here, vectors are in D-dimensional space, where D = 100.

N = number of words in description 1 (Total: 23)

v1 = vector representation of job description

This is how we calculate the average Word2vec. In the same way, the other job descriptions can be converted into vectors.

B. TF-IDF Vectorization

By this model, the similarity between two datasets can be found by determining cosine similarity value between two vectors. In case of recommendation engines, the similarity value between user query and documents are determined and then it is categorized from highest to lowest one. Cosine similarity on the other hand still can't deal with the semantic meaning of the query very well. Semantic meaning problem does not meet the difference of syntax matching.

Besides, As previously mentioned the KNN algorithm is based on machine learning. Preprocessing and document preparation is followed by the learning phase. The algorithm determines the basic documents which will be compared with each new document. Algorithm check where a document is categorized by only looking at the training documents that are most similar to it.
The KNN algorithm assumes that it is possible to classify documents in the Euclidean space as points. Euclidean distance is the distance between two points in Euclidean space. The distance between two points in the plane with coordinates p= (x, y) and q= (a, b) can be calculated [19]:

$$D(p, q) = d(q, p) = \sqrt{(x - a)^2 + (y - b)^2} \quad \dots\dots\dots\dots\dots\dots (9)$$

C. TF-IDF*Word2Vec

This method is a combination of method A and B. To elaborate the calculation, assume that a job description has 23 words. Let's denote the words as w1, w2, w3, w4 …w23.
*Steps to calculate the TF-IDF*Word2Vec:*

i. Calculate the TF-IDF vector for each word in the above description. Let's call the TF-IDF vectors as tf1, tf2, tf3, ..., tf23. Please note TF-IDF vector won't give D-dimensional vectors
ii. Calculate the Word2Vec for each word in the description
iii. Multiply the TF-IDF score and Word2Vec vector representation of each word and total
iv. Then divide the total by the sum of TF-IDF vectors. It can be called v1 and written as follow

$$\frac{tf1 * W2V(w_1) + tf2 * W2V(w_2) + \dots.. + tf23 * W2V(w_{23})}{tf1 + tf2 + tf3 + \dots.. + tf23} \quad \dots\dots\dots\dots\dots\dots (10)$$

v1 = vector representation of jobs description 1. This is the method for calculating TF-IDF Word2Vec. In the same way, we can convert the other job descriptions into vectors.

EXPERIMENTATION AND RESULTS

We have used seven different methods to evaluate the performance of our word2vec model constructed based on this data. For testing purposes, a single tuple of doctors' dataset has been used. The first method was average word2vec within cosine similarity. The early mentioned test data was fed into the average word2vec model then compared similarity among all features of the job dataset and top 5 job features id in this case job_code was displayed according to their similarity values. The obtained job features within their similarity values are given below:

Top-Similarity	Similarity Values	Job Code
1	0.880896671378053	52670-j21
2	0.8751919678434482	W50618
3	0.8682868045339545	W55015
4	0.8620885517763219	08698-j13
5	0.8564568969436167	W50561

Table 1: The obtained job features within similarity values from the average word2vec within cosine similarity model

The second method was TF-IDF Vectorization within cosine similarity applied in similar way with similar test data and the obtained outcome are as follows:

Top-Similarity	Similarity Values	Job Code
1	0.4237582826260231	52306
2	0.4017369227696345	08775
3	0.4017369227696345	08775-j42
4	0.31534308121781796	W03406
5	0.3052636274348004	W51007

Table 2: The obtained job features within similarity values from the TF-IDF Vectorization within cosine similarity model

In the third approach TF-IDF vectorization was combined with average word2vec for constructing word vectors then compared this list of vectors among all features of the job dataset through cosine similarity. The obtained outcome of this method are as follows:

Top-Similarity	Similarity Values	Job Code
1	0.8261491210516927	W50618
2	0.8254814138323184	02834-j21
3	0.8204593421907131	W55183
4	0.8174754839293676	W02920
5	0.8153469767508703	W55175

Table 3: The obtained job features within similarity values from the combined TF-IDF Vectorization and average word2vec model within cosine similarity model

Ensemble method of above three was used as the fourth method which has been implemented by averaging all cosine similarities values then searching related job features based on those values and in similar way only top 5 job indices were displayed. Here are the obtained results:

Top-Similarity	Similarity Values	Job Code
1	0.6507160402338046	52306
2	0.6275769749759949	02834-j21
3	0.6197157315296643	06005-j21
4	0.6174876105621611	W03406
5	0.6113819123253353	W04573

Table 4: The obtained job features within similarity values from the Ensemble model

Next average word2vec method within KNN was used as the fifth method where ten nearest neighbours of the test feature were obtained. Here are they:

Top-Neighbors	10 Nearest Distances	Job Code
1	1.1790209314182298	52670-j21
2	1.179439738454561	W50618
3	1.2197959476689622	W55015
4	1.2708236816266065	08698-j13
5	1.2721959745476086	W50561

6	1.2818139914666788	W55489
7	1.285656778449799	09213-j04
8	1.2862622798197263	W50179
9	1.294702313894636	09907-j17
10	1.304742515395499	06005-j21

Table 5: The obtained job features within similarity values from the average word2vec model within KNN

After that KNN was used with other two methods in similar ways as the sixth and seventh method. The TF-IDF vectorization within KNN was used as the sixth method and the combination of TF-IDF vectorization and average word2vec was used within KNN as the seventh method. The obtained results of those two methods are given below respectfully:

Top-Neighbors	10 Nearest Distances	Job Code
1	0.9999999999999999	10866-j42
2	0.9999999999999999	51791-j42
3	0.9999999999999999	52346-j42
4	0.9999999999999999	01070
5	1.0735378124444213	52306
6	1.0938583795266785	08775
7	1.0938583795266785	08775-j42
8	1.1701768402956727	W03406
9	1.1787589851748317	W51007
10	1.181859651993979	56232

Table 6: The obtained job features within similarity values from the TF-IDF vectorization model within KNN

The last one:

Top-Neighbors	10 Nearest Distances	Job Code
1	1.4133246843831817	W50618

2	1.4578297361160675	W55175
3	1.4591515291101285	W00164
4	1.4628999484620908	51895
5	1.4711947169918342	02834-j21
6	1.4856914212023904	05818-j30
7	1.4933904849796065	W54666
8	1.4942091570074107	W04056
9	1.4975597410923858	W55183
10	1.5053673665951346	W55432

Table 7: The obtained job features within similarity values from the combined TF-IDF Vectorization and average word2vec model within KNN

Performance Evaluation

For performance evaluation obtained job features were compared with actual job features using cosine similarity after extraction using obtained index number which is in this case job code. The first and most similar job code of the first method was '52670-j21' and in comparison, of actual similar job features return cosine similarity value only 24.28% which is not very reliable. The other similarity values from the comparison among the actual features and the predicted features of this method are 8.54%, 5.88%, 10.41%, 5.91% respectively which are comparatively very poor from the first one. The corresponding values of this performance measure is given below:

Obtained Job Code	Relevant Indices	Similarity (100%)
52670-j21	24296	24.282146558931604
W50618	31499	8.539125638299666
W55015	323205	5.879505427397483
08698-j13	20999	10.416666666666668
W50561	31442	5.905963936799736

Table 8: The performance measurement of the average word2vec within cosine similarity model

The second method, which is the TFIDF vectorization within cosine similarity which provides better performance than the previous one. Similarity measures between the predicted output obtained from this method and the actual features returned these values respectively 31.62%, 55.33%, 40.82%, 22.36% and 26.22%. But still this method is not also a reliable one since only one value has more than half chance to be similar. The performance measurement of this model is given below:

Obtained Job Code	Relevant Indices	Similarity (100%)
52306	9825	31.622776601683793
08775	6309	55.339859052946636
08775-j42	21048	40.8248290463863
W03406	29089	22.360679774997898
W51007	31759	26.726124191242434

Table 9: The performance measurement of the TF-IDF Vectorization within cosine similarity model

The similarity measure for the third method's obtained results are 8.54%, 21.08%, 2.8%, 9.07% and 2.43% and which are very less and by observing this value it certainly concludes that this model is not appropriate which means average word2vec could not be combined with TF-IDF vectorization in this way. Here it is:

Obtained Job Code	Relevant Indices	Similarity (100%)
W50618	31499	8.539125638299666
02834-j21	18011	21.08185106778919
W55183	32489	2.7950849718747373
W02920	29241	9.068453126375147
W55175	32481	2.4282146558931603

Table 10: The performance measurement of the combined TF-IDF Vectorization and average word2vec model within cosine similarity model

The 4th model, which is an ensemble model, provides those values in performance measures: 31.62%, 21.08%, 22.43%, 22.36% and 15.81%. This model also couldn't bypass the TF-IDF vectorization within cosine similarity. The performance measurement values of this model are given below:

Obtained Job Code	Relevant Indices	Similarity (100%)
52306	9825	31.622776601683793

02834-j21	18011	21.08185106778919
06005-j21	19200	22.42713067862651
W03406	29089	22.360679774997898
W04573	29751	15.811388300841896

Table 11: The performance measurement of the Ensemble model

Next the performance evaluation of the all KNN models. KNN within average word2vec provides similarity values in performance measurement: 24.28%, 8.54%, 5.88%, 10.42%, 5.91%, 6.7%, 5.83%, 9.83% and 22.42%. The notable fact of this method is that out of 10 individual outcomes, 9 job features were found in actual data; one job code 'W55489' was not present in this dataset; hence this value could be considered an outlier regarding this dataset. Therefore, though it didn't provide any mentionable performance it's worthy in outlier detection. The performance measurement values of this method are shown in following table:

Obtained Job Code	Relevant Indices	Similarity (100%)
52670-j21	24296	24.282146558931604
W50618	31499	8.539125638299666
W55015	32320	5.879505427397483
08698-j13	20999	10.416666666666668
W50561	31442	5.905963936799736
W55489	No Index Found	
09213-j04	21287	6.694827640161777
W50179	31094	5.829915557523537
09907-j17	21783	9.829463743659808
06005-j21	19200	22.42713067862651

Table 12: The performance measurement of the average word2vec model within KNN

Next the TF-IDF vectorization within KNN method: 17.68%, 17.68%, 17.68%, 17.68%, 31.62%, 55.33%, 40.82%, 22.36%, 26.73% and 14.14%. This method doesn't provide any outlier and also returns a job feature which provides 55.33% similarity value. Here is the orderly representation of these values:

Obtained Job Code	Relevant Indices	Similarity (100%)
10866-j42	22361	17.677669529663685
51791-j42	28509	17.677669529663685
52346-j42	24017	17.677669529663685
01070	785	17.677669529663685
52306	9825	31.622776601683793
08775	6309	55.339859052946636
08775-j42	21048	40.8248290463863
W03406	29089	22.360679774997898
W51007	31759	26.726124191242434
56232	12990	14.142135623730951

Table 13: The performance measurement of the TF-IDF vectorization model within KNN

The last method, TF-IDF and average word2vec combination within KNN's performance values are 8.54%, 2.43%, 4.32%, 2.42%, 4.28%, 2.8% and 4.08%. This model is also not reliable since its performance is not even up to 10% and it also provides 3 garbage job features value which are not present in actual data. The orderly representation of the performance of this model is as follows:

Obtained Job Code	Relevant Indices	Similarity (100%)
W50618	31499	8.539125638299666
W55175	32481	2.4282146558931603
W00164	28534	4.3193421279068005
51895	No Index Found	
02834-j21	No Index Found	
05818-j30	No Index Found	
W54666	32294	2.416841222614159
W04056	29460	4.27960492510913
W55183	324899	2.7950849718747373

W55432	32746	4.079462199153158

Table 14: The performance measurement of the combined TF-IDF Vectorization and average word2vec model within KNN

From above seven methods two methods which are TF-IDF with average word2vec combination within cosine similarity and same combined model within KNN are totally unreliable in both the relevant results case as well as performance case hence we could reach in this conclusion for this study that this type of combination couldn't be done. Among rest of the five methods TF-IDF with cosine similarity along with same model with KNN both provides better performances than others still not reliable performance but yet this opens an opportunity for further research because there is a possibility this performance could be increased through parameter tuning or feature changing or feature extending hence this type of model could be accepted. The two other models based on average word2vec couldn't be reliable for this dataset but could be experimented on other dataset or study. And the remaining ensemble method neither provides better performance than the TF-IDF based model nor poorer than the average word2vec based model but further experiment could be performed on this one also like parameter tuning or feature tuning. Though this study couldn't provide a reliable outcome, it opens the door for further analysis also identifying some inappropriate approaches which are the best findings from this study.

CONCLUSION

This study was an attempt to build a recommendation engine for a job searching website for doctors. Three different vectorization algorithms were sequentially executed with KNN and cosine similarity. Also, a new model created by averaging three different cosine similarity values obtained from the sequential cosine similarity models with three different vectorization algorithms. Then test the relevancy through performance measuring algorithms. This performance measuring algorithm provides each model's performance individually which was previously shown in performance evaluation. It reveals two inappropriate models which were created by combining TF-IDF with Average Word2vec then sequentially executed with KNN and cosine similarity. It enhances that combining those two vectorization algorithms was not a good idea which will restrict researchers from making this type of combination. It's a novel finding from this study. For the other seven models it reveals the most appropriate model for this dataset and also open further research concepts such as checking those models with other dataset and find the appropriate model for that dataset and also hyper parameter optimization or feature exchanging or feature extending to increase performance of the most appropriate model which was the TF-IDF model with cosine similarity.

REFERENCES

1. Xiao, Bo, and Izak Benbasat. “E-Commerce Product Recommendation Agents: Use, Characteristics, and Impact.” MIS Quarterly, vol. 31, no. 1, 2007, pp. 137–209. JSTOR, www.jstor.org/stable/25148784. Accessed 10 June 2021.
2. Bond, F., & Baldwin, T. (2016). Introduction to Japanese Computational Linguistics. Nakagawa, H., & Shimazu, A. (Eds.), Readings in Japanese Natural Language Processing, 1-28.
3. Beel, Joeran et al. “Research-paper recommender systems: a literature survey.” International Journal on Digital Libraries 17 (2015): 305-338.
4. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: 26th Int. Conf. on Neural Information Processing Systems. pp. 3111–3119 (2013)
5. Mikolov, T., Yih, W.t., Zweig, G.: Linguistic regularities in continuous space word representations. In: HLT-NAACL. pp. 746–751 (2013)
6. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: EMNLP. vol. 14, pp. 1532–1543 (2014)
7. Dridi A., Gaber M.M., Azad R.M.A., Bhogal J. (2018) KNN Embedding Stability for word2vec Hyper-Parameterization in Scientific Text. In: Soldatova L., Vanschoren J., Papadopoulos G., Ceci M. (eds) Discovery Science. DS 2018. Lecture Notes in Computer Science, vol 11198. Springer, Cham.
8. Kudo, T., Yamamoto, K., and Matsumoto, Y. (2004). Applying conditional random fields to japanese morphological analysis. In Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 230–237.
9. http://taku910.github.io/mecab/
10. https://www.atilika.com/ja/kuromoji/
11. Y. Bengio, R. Ducharme, P. Vincent. A neural probabilistic language model. Journal of Machine Learning Research, 3:1137-1155, 2003.
12. T. Mikolov. Language Modeling for Speech Recognition in Czech, Masters thesis, Brno University of Technology, 2007.
13. T. Mikolov, J. Kopecky, L. Burget, O. Glembek and J. ´ Cernock ˇ y. Neural network based lan-´guage models for highly inflected languages, In: Proc. ICASSP 2009.
14. T Mikolov et al. “Efficient estimation of word representations in vector space”. In: arXiv preprint arXiv:1301.3781 (2013).
15. Husain, Wahidah & Dih, Lam. (2012). A Framework of a Personalized Location-based Traveler Recommendation System in Mobile Application. International Journal of Multimedia and Ubiquitous Engineering.
16. Yoav Goldberg, Omer Levy (2014). word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method.

17. https://radimrehurek.com/gensim/models/word2vec.html
18. Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso (2015). Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.
19. McInnes, L, Healy, J, UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, ArXiv e-prints 1802.03426, 2018
20. M Diaby, E Viennet, and T Launay. "Toward the next generation of recruitment tools: An online social network-based job recommender system". In: Proc. of the 2013 IEEE/ACM Int. Conf. On Advances in Social Network.
21. G Salton and C Buckley. "Term-weighting approaches in automatic text retrieval". In: Information Processing and Management 24.5 (1988), pp. 513–523. issn: 0306-4573. doi: https://doi.org/10.1016/0306- 4573(88)90021- 0. url: http://www.sciencedirect.com/science/article/pii/0306457388900210.
22. Ming-Yang Su, using clustering to improve the KNN-based classifiers for online anomaly network traffic identification, Journal of Network and Computer Applications 34 (2011) 722–730.