DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Topic Modeling for Employee Objectives using Word Embeddings

Anum Afzal

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Topic Modeling for Employee Objectives using Word Embeddings

Themenmodellierung für Mitarbeiterziele mit Word Einbettungen

Author:	Anum Afzal
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Ahmed Elnaggar
Submission Date:	28.12.2020

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 28.12.2020 Anum Afzal

Acknowledgments

First of all, I would also like to thank Prof. Dr. Florian Matthes who gave me an opportunity to work with his chair. Secondly, I want to offer my gratitude to my thesis advisor Ahmed Elnaggar for his guidance, patience & motivation throughout the thesis. This thesis would not have gone this smoothly without his active participation and I couldn't have asked for a better advisor. Thirdly, I want thank Merck Group for sharing this project with us and also providing feedback on the final implementation.

Next, I would like to thank my parents for their sacrifices and endurances. My brother Naveed for introducing me to Computer Science and providing academic advises. My sisters Nimra and Fariha for their support and love.

I also want to express my appreciation towards all my dear friends who are listed in alphabetical order. Ahsan for always being there for me irrespective of the time and day. Asima for being a great friend and baby sitting me while I wrote this document. Mastaneh who helps me put things into perspective. Nicholas for always telling me what I need to hear and also serving as the one-man Quality Assurance team for this project. Ons for always encouraging me and being a great listener. Sandhya for always believing in me and being a life-long friend. Suleman who also believed in me no matter what and always pushed me to excel.

In the end, I also want to thank all the people who reviewed the thesis and provided their feedback. Especially my brother Naveed, Asima and the ever lovely Zoha.

Abstract

Classical Machine Learning approaches for Natural Language Processing (NLP) work well for most cases but there is always a threshold in terms of accuracy which can never be crossed. On the other hand, state-of-the-art Deep Learning models have managed to surpass that threshold and come very close to human accuracy. This study focuses on performing a Topic Modelling task using Deep Learning models and comparing results with a classical approach known as Latent Dirichlet Allocation.

Topic Modeling is an unsupervised Machine Learning technique that groups documents into clusters and finds topic words for each cluster. At Merck Group, Topic Modeling is used to understand the objectives of employees without reading the documents. The general idea is to group employees into clusters based on the similarity of their objectives and find topics which depict the main goals of the cluster's employees.

While an LDA model is able to provide good results, it has certain limitations. First, it works purely on the frequency of words in a document, and all stop-words are removed as a part of the pre-processing. This leads to loss of information in terms of grammatical context and order of words in a sentence. Secondly, it is common for documents to have different words with the same meaning. In an LDA model, these same meaning words would be treated as different.

A word embedding model provides a solution to the above-mentioned problems as it is able to retain all grammatical information by processing the sentence as a whole. Additionally, a Word Embedding model provides a multi-dimensional representation for each word which allows capturing the contextual similarity of words with the same meaning.

This study demonstrates how using the feature vectors from an Embedding Model, and more specifically, a Sentence Embedding model can provide better results than an LDA model. This study also discusses various Topic word retrieval techniques and concludes that the frequency-based approach and TF-IDF provide the most coherent topic words. Lastly, it also discusses that analytical measures such as the Silhouette score and Coherence score are not suitable for Topic Modeling.

Contents

1 Introduction

1.1 Overview

Deep Learning has slowly found its application in almost all research areas such as Computer Vision, Bio-medicine, Financial sectors, etc. People Analytic is one of emerging application of Deep Learning which uses deep learning algorithms to better manage the employees in a company. Many HR practitioners are trying to bring in state-of-the-art Deep Learning approaches to understand the goals and needs of employees to maximize their satisfaction and productivity.

Employee management is indeed a very challenging task, especially for big companies, from understanding the needs of individuals and employee retention to aligning the goals of employee to that of the company. Usually companies use manual tools and techniques to understand the objectives of an employee but this is a very time consuming task. In order to optimize this, many companies are moving towards Artificial Intelligence tools which does this for them automatically. Merck Group being a multi-national company is also working on Deep Learning approaches to better understand the self-evaluation objectives written by the employees.

1.2 Motivation

At Merck Group, it is crucial for a managerial level individual such as the CEO or HOD to have an understanding of the employee objectives. However, given their busy schedule, it is very challenging and time-consuming for them to analyze the objectives of hundreds of employees. Moreover, this process has to be repeated every 6 months. To target this problem, Merck Group is interested in having a tool which analyzes the objectives, extracts conclusions using deep learning techniques and visualizes the results through a Graphical user interface. With a tool like this, companies such as Merck Group would be able to understand the goals and needs of all employees as a whole as well as on an individual level.

There are certain challenges in designing such a tool. For any machine learning problem, it is important to have good quality labeled data. However, in reality it is often times not easy to find labeled data. Given the digital age, there are many available sources of unstructured data which contains a huge ensemble of information. Such data makes it almost impossible to train a new model or fine-tune an existing one. This is also the case for this study as the Employee Objective dataset is unlabeled and unstructured but it still contains useful information. Dealing with such a dataset is still challenging as there is no prior knowledge available about it such as the number of employee clusters or the ground truth to compare

the results with. Fortunately, Scientists have come up with techniques that are able to extract information from unstructured data. Unsupervised learning techniques such as Topic Modeling is one of them.

1.3 Problem Statement

Topic Modeling is a branch of Natural Language Processing (NLP) that provides an efficient solution to the problem of analyzing Employee Objectives. Classical NLP technique such as Latent Dirichlet Allocation (LDA) [1] provides a good solution to this problem but has two major drawbacks. The first drawback is the loss of information through stopwords removal and stemming. An LDA models need a token list for each document as an input. Since the stop words do not play an important role in the model training, they are removed from the token list. Furthermore, to obtain better results, the items in the token lists are stemmed. The grammatical form of words which contains a lot of useful information is lost. Moreover, since an LDA model is only interested in the frequency of words, the order in which the words appear in a sentence is also disregarded. Second drawback of LDA is that it's working Principle operates on the occurrence of a word rather than the contextual meaning in which it is used. It essentially assumes that the words which appear together must belong in the same group. If the words 'good' and 'bad' exists in each other's company quite often, the model assumes that they belong together. This is true for the case when they are both adjectives and define a quality but it is false on premise that one defines a positive quality and the other defines a negative quality.

There are a lot of gray areas when dealing with human languages as the words have different meanings, depending on the context they are used in. There are several words which could have the same meanings and one word could have multiple meanings. This degree of variability is not captured by an LDA model which is probabilistic and deterministic in nature. To deal with human languages effectively, it is essential to work with a model that is able to capture the meanings of words along multiple dimensions. Fortunately, Deep Learning models for Natural Language Processing are able to provide a solution for this by training word embedding along multiple dimensions.

1.4 Research Questions

A Word Embedding model provides a matrix representation for text allowing it to capture the context and semantic of each word across many dimensions. Since the embedding models take the raw text as an input, no information is lost in terms of grammar and stop words. This provides more rich vector representation by utilizing the information obtained through stop words, grammatical forms of words and the order in which they appear. The embedding vectors provide a multi-dimensional [2] representation for each word in a text such that words could be similar to each other and lie close to each other in one domain while be different from each other and lie far away from each other in another domain. Hence, using such a

representation could enable the model to treat different words with similar meaning as same and also capture multiple meanings of the same word.

A lot of this makes sense in theory but it is also important to have some results to prove it. This study focuses on three research questions.

1. Could using embedding vectors lead to better results than Latent Dirichlet Allocation model?
2. If the word embedding models are able to provide better results, then which type of embedding model is better suited?
3. Could using a traditional algorithm such as LDA in tandem with the Embedding models provide better results?

1.5 Research Contributions

The outcome of this study is a flask-based web applications with a Graphical User Interface (GUI) and an NLP engine at the back-end. The GUI provides options to select the type of visualizations, feature space, and number of clusters, among other hyperparameters to visualize the employee clusters and their topic words. The NLP engine is fine-tuned using the findings from the experiments section. This study compares results from various types of word embedding models as BERT [3], Sentence Bert [4], XLNET [5] among others with LDA model and concludes that Sentence Bert provides the best results. Furthermore, it also experiments with latent features spaces obtained using an Autoencoder. This study also contributes by presenting results for various configurations of topic word techniques.

1.6 Research Approach

The research approach followed in this project is a Deductive one which given an existing theory tries to develop a new hypothesis and also tests it [6] through experimentation. The given theory in this study is that traditional Topic Modeling approach such as LDA fail to capture the context of the text and is also subject to loss of information. It is also known that word embedding models are designed to overcome these issues. Therefore, this research tries to prove a new hypothesis that results for topic modeling could be improved by incorporating the feature vectors from the embedding models. This hypothesis is tested on the employee objective dataset provided by the Merck Group as well as on an open source Job Description dataset from Kaggle.

1.7 Structure

The document is divided into ten chapters. First chapter "Introduction" which is this chapter. The second chapter is "Background" which explains the basics needed the understand the later chapters. Chapter three is "Related Work" that provides an summary of other researches

in this area. Chapter four "System Architecture" explains the communication between the front-end and the back-end of the flask application. Chapter five is the "Methodology" section that discusses the intuition behind the algorithms and techniques used in this study. Chapter six "Datasets" provides an explanation as well as statistical analysis of the datasets used for experimentation. Chapter seven "Experiments" provides details of the configurations used for the algorithms and techniques defined in the methodology chapter. The results from the experiments are discussed in Chapter eight "Results". The study is concluded in chapter nice "Conclusion" and possible improvements and build-ups are discussed in chapter ten "Future works".

2 Foundations of Deep Learning and Natural Language Processing

This chapter focuses on building the basics of Natural Language Processing, Deep Learning, and Clustering. The first section explains the basics of Natural Language Processing and introduces some classical techniques for dealing with unstructured data. It discusses the pre-processing steps involved before the data can be fed to the model and also the challenges of working with languages [7]. The second section gives a detailed explanation on the basics and working of Neural Networks followed by the third section which discusses some deep learning architectures meant for Natural Language Processing. The last and forth section of this chapter gives an insight into unsupervised learning approach called Clustering and discusses K-mean clustering algorithm in detail.

2.1 Natural Language Processing

The most effective form of communication are the words we use. Either written or spoken, they carry a lot of information. In this age of digitization, researchers are working on techniques that allow models to effectively process human languages. This is a very difficult task as the human understanding of a text incorporates a lot of information such as the spoken tone and the context in which the sentence was written or spoken and even the emotions behind it. Natural Language Processing (NLP) stems as a branch of Artificial Intelligence (AI) that gives machines the ability to not only read but also understand languages. While earlier algorithms focused on classical techniques that are probabilistic and deterministic in nature, state-of-the-art deep learning model these days are able to understand text on a more contextual level. Some state-of-the-art language models are explained in [8] and also discussed in the section 2.3 of this chapter. What makes human languages more complex than other tasks is the concept of temporality [9]. In a basic dataset, the features in a sample are not very dependent on each other. For example, In a housing example, the order of features 'Number of rooms' and 'median income' does not matter. The model training would have no effect if 'median income' features is listed before 'Number of rooms'. This is not the case of NLP data, there is a degree of dependency between words in a text and shuffling the tokens (words) before passing to the model can heavily effect the model training. For example, the sentence 'My name is John' would have a different effect in model is it is passed as 'Name is my John'. Another important factor is that the text cannot be directly fed to a model and has to be converted into a numeric form which is understood by a model. Below are the steps that convert text into a machine recognizable form.

1. The basic idea is to first break the sentence into tokens. So the sentence 'My name is John' would be tokenized to ['My', 'name', 'is', 'John']

2. create a dictionary that has all the words used in the text and assign each word an id. For example 'My': 0, 'name': 1, 'is': 2, 'John': 3 and replace the tokenized list of words with their respective ids such as [0, 1, 2, 3]

3. Step 1 and Step 2 are repeated for all the text in the dataset. If the word already exists in the dictionary, then the existing id is used instead of assigning a new one.

NLP is a very broad field addressing many text related tasks. Some of the traditional tasks relevant to this study are mentioned below:

2.1.1 Unstructured Data

Unstructured data [10] is information that doesn't fit into the traditional columns and rows. It has no apparent description or length. In this digital age, there are a lot of sources of unstructured data such as emails, tweets, blog posts, and reviews. However, the issue is that the data is in raw form and since most machine learning techniques require structured data with hand crafted features, it was of no use until the NLP community figured out ways of utilizing this information. This study deals with unstructured data while focusing on clustering similar texts and finding the keywords for each cluster. Hence, these two tasks are discussed in more details below:

Text Clustering

Natural Language Processing is one of the oldest and versatile research area in computer science covering applications such as Chatbots, Next Sentence Prediction, Language Translations, Sentiment Analysis and many more. However, for most of these applications, it is necessary to have labelled and structured data. Finding a good quality labelled data is quite difficult in a real world setting. Text clustering or Document clustering [11] is a powerful technique for getting insights out of unlabelled data. Text clustering follows three steps:

1. Pre-processing: Text is converted to numeric form. This process typically involves word tokenization and stemming

2. Feature Extraction: This can be done by either using a Bag-of-words representation where the frequency of each word in a text is counted and passed to the model as features. Another option is to calculate the TF-IDF scores for each text and use as features.

3. Clustering: Based on the features, similar documents are grouped into same clusters. This requires a distance measure to calculate the similarity of the documents. Less distance between documents means they are more similar.

There are two clustering policies; Hard clustering and soft clustering. In hard clustering, the document can only belong to one cluster whereas in soft clustering, the document can belong to more than one cluster as shown in figure 2.1.

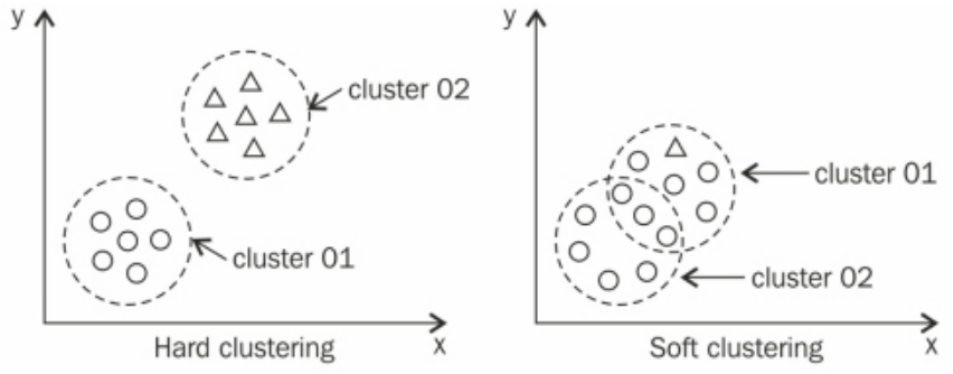


Figure 2.1: Hard vs Soft clustering policy.

2.1.2 Keyword Retrieval

Keyword Retrieval [12] [13] [14] algorithm work by finding important keywords in a given text. These techniques mostly operate on the frequency of word appearance in the text which work well in a simple setting. But one drawback associated with them is that there is no notion of context involved in them. Some state-of-the-art deep learning based algorithms use the attention mechanism of transformer models to find the important words in a text. Using the attention [15] mechanism it is possible to also incorporate the context from the surround words while finding important words. This technique is discussed in more details in [15]. Nonetheless, the classical approaches for keyword retrieval are very powerful and simple techniques and it is important to discuss them. TF-IDF is the most commonly used technique for finding important keywords which is also used in this study and discussed below.

Term Frequency-Inverse Document Frequency (TF-IDF)

When dealing a collection of texts, TF-IDF helps find important and unique words in it. TF-IDF [16] is a statistical algorithms that works in two parts; the first parts finds the importance of a word by measuring it's frequency in the text. Even though, the frequency of a word can be a good indicator of a word's importance, it can be misleading sometimes. For example, stopwords such as 'is', 'a' 'which' often times have a higher frequency than the other words. One option of dealing with it is to manually filter out all the stopwords but TF-IDF algorithm provides an automatic way of dealing with it. The second part of the algorithm IDF checks if the word also occurs very commonly in all other texts, if so then the algorithm decreases the importance of those words. In a nut shell, by combining these two techniques together, TF-IDF gives keywords that are important to each text and also unique with respect to the rest of texts.

2.2 Deep Learning

Deep Learning is a Machine Learning technique that automatically learns the feature of the data. It performs better than all other machine learning algorithm as it doesn't require hand-crafted features like most Machine Learning models. Additionally, deep learning is quite scalable and it has found it's applications in almost all research fields such as Computer Vision [17] [18], Natural Language Processing [19] [20], Bio-medicine [21] [22] [23] and many more. The later sub-sections of this chapter will provide some basic background of deep learning and then discuss some state-of-the-art models specific to Natural Language Processing.

2.2.1 Introduction

The most basic working principle of a deep learning models is to replicate the learning behaviour of a human brain. For this reason, deep learning models are often referred to as an Artificial Neural Networks. Just like how a brain has billions of neurons which are connected by synapses, an artificial neural network has nodes and layers which try to depict the function of neurons and synapses respectively. Just like how the signal travels from one node to another through synapses, a neural network has hundreds of layers interconnected with each other to facilitate the flow of information within the model. A typical neural network can be broken down into three parts. The first layer which is known as the input layer, followed by hundreds of hidden layers where each layer can have arbitrary number of nodes. The last layer in the output layer which provides the final model prediction. All the nodes of all the layers between the input and output are connected to each other which represents the flow of conversation happening between the nodes. Each nodes receives an input from it's previous layer and after doing some computations on it, passes it to the nodes in the next layers. These infinitely many connections between the nodes is what makes these model so powerful in the first place as it allows the model to learn complex features of the data. A very basic illustration of a neural network can be seen in figure Figure 2.2

2.2.2 Basic of an Artificial Neural Networks

In order for a neural network to produce accurate results, it needs appropriate weights. Essentially any input that is given to the model, passes through all the layers of the models. This mechanism is known as the forward pass in which the input is multiplied to the weights of all the layers. Hence, before a model can provide good predictions, it must go through a learning phase where it adjusts it's weights by looking at the input samples.

To get a better understanding of how a neural network works, it might be helpful to first understand what happens to a single node in a layer. As shown in Figure 2.3, a neuron(node) receives the features x_1, x_2, x_3, ..., x_n from the previous layers. These features are then multiplied by the weights w_1, w_2, w_3, ..., w_n and passed through a summation to get a single value as the output. As a standard practice, this output is passed through an activation function [24] which in most cases adjusts the output between 1 and 0, or -1 and +1, or 0 and

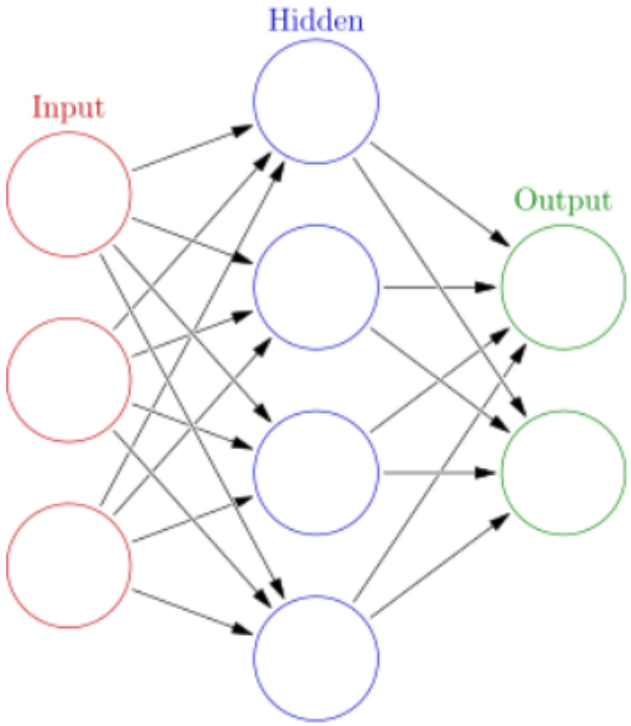


Figure 2.2: A neural network with only one hidden layer.

positive values. The choice of the activation function varies of each model and highly depends on the underlying application. Some commonly used activation functions are discussed in the section 2.2.2

Looking at the bigger picture, each node in a layer receives the output of all the nodes from the previous layer as it's input. Similarly, the output of each node is passed to all the nodes in the layer next to it. A model with hundreds of layers with hundreds of nodes for each layer is able to perform very deep and nested calculations and is known as a Deep Neural Network.

The weights of a network are the backbone of the whole model therefore, before moving forwards it is important to understand what they actually mean. The weights of a model are basically matrices containing numbers which are initialized with random values in the beginning. During training, the models sees new samples as inputs and keep adjusting the weights accordingly. The idea is to keep training and changing the weights until the output is good enough. Two important concepts involved in the weights update are the loss function and the concept of back-propagation. Both of them along with the activation functions are explained below.

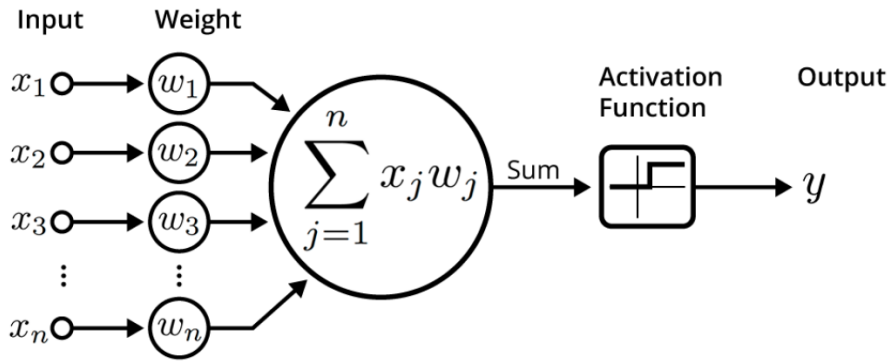


Figure 2.3: A sample neuron inside a neural network.

Activation Functions

Some common choices of activation functions includes the following:

1. Threshold Functions: This is the most basic form of activation function. As the name suggest , if input value is above the threshold value, it return 1 and if it below the threshold value, it returns 0. Most often variants of a threshold function is a unit step function that return 1 for value zero and above and returns 0 for negative values. The graph for a unit-step function can be seen in figure 2.4

2. Sigmoid Functions: This is the most commonly used activation function in the deep learning community. For an given input, a Sigmoid [25] function returns a value between 0 and 1. This works very well with most models as it gives a probabilistic measure as a output. One example of such a case would be a Binary Classification model. Given an email, the model needs to predict if it is spam or not. Instead of treating it as black and white, it is a better practice to give a probabilistic measure and conclude that the email is 60% spam. The graph for a sigmoid function can be seen in figure 2.5

3. Rectifier Functions: This function is most commonly known as a ReLU [26] function and is used to avoid negative inputs. For an input value, it returns a 0 if it is negative otherwise it outputs the input as it is. The graph for a relu function is shown in figure 2.6

4. Hyperbolic Tangent Functions: This function is commonly know as a tanh function because of the trigonometric identity involved. It works similarly to a sigmoid function but has a different range of output. For any input, it returns a value between +1 and -1 as shown in figure 2.7

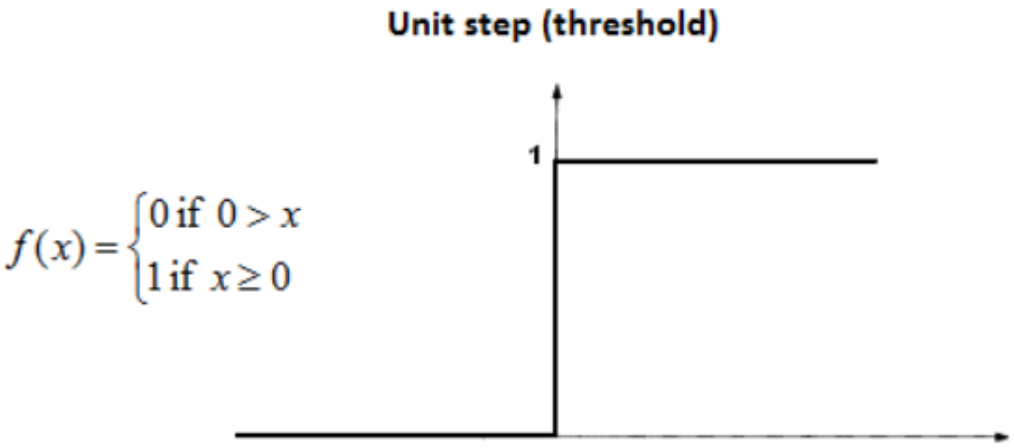


Figure 2.4: The mathematical form and graph for a threshold function

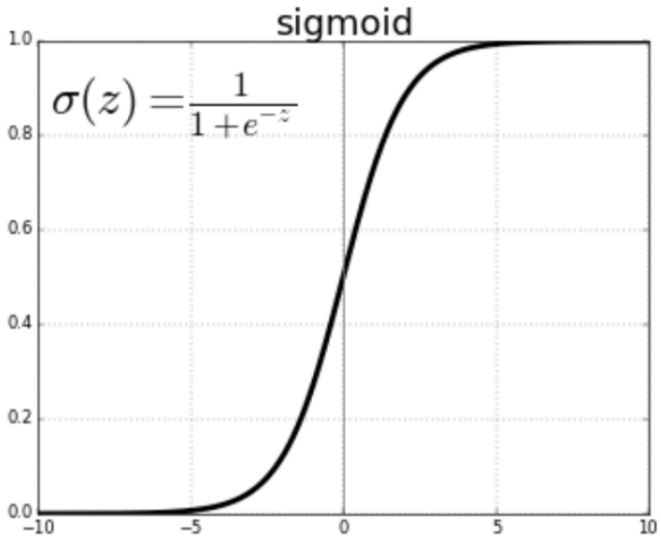


Figure 2.5: The mathematical form and graph for a sigmoid function

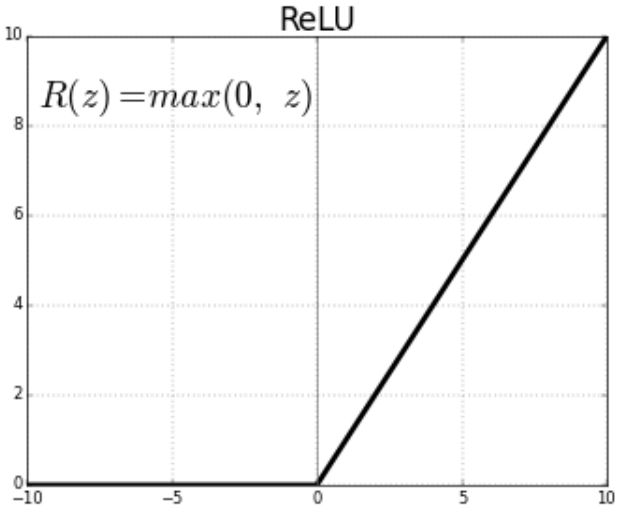


Figure 2.6: The mathematical form and graph for a reLU function

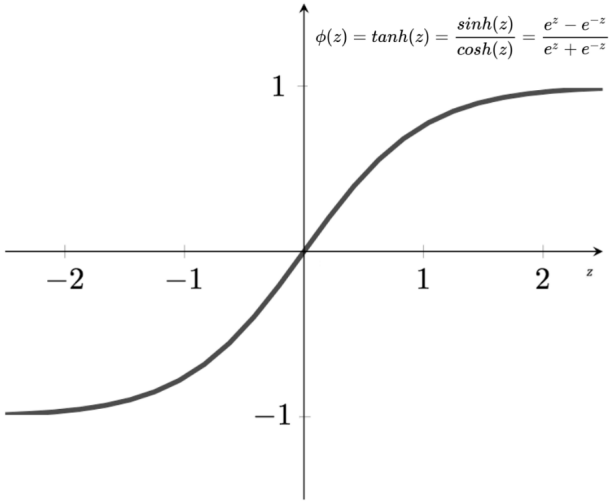


Figure 2.7: The mathematical form and graph for a tanh function

Loss Function

A loss function is essentially a mathematical formula that calculates the difference between the model's predicted value and the actual output. This loss values does not only provide a measure to evaluate the models performance but also plays a vital role in the weight estimation happening through back propagation which is mentioned in the section 2.2.2 A loss function is picked depending on output requirement of the model. Some common choices of loss functions are discussed below:

1. Mean Squared Error Loss In the simplest form, a mean squared error loss (MSE) is defined as $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ where y_i is the actual output for a sample i and $\hat{y}_i$ is the model prediction for sample i. The MSE loss functions takes the difference between the actual and predicted values for all samples and takes to mean to get the loss for the whole dataset. This loss function works well for a regression model where the outputs are the real values.

2. Cross-Entropy Loss An MSE loss function doesn't work well with a classification problem where the output needs to be 0 or 1 rather than real numbers. For such a case, the formula to calculate the loss is $-(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i))$ where y_i is the actual output for a sample i and $\hat{y}_i$ is the model prediction for sample i. This loss function works in two parts.

1. First is for the case when $y_i = 1$, the second term of the formula disappears as $1 - y_i$ become 0. In this case, the loss function simply multiplies the log of predicted value to the actual value.

2. The second case occurs when $y_i = 0$, the first term of the formula disappears as $y_i = 0$. In this case, the loss function also multiplies the log of predicted value to the actual value.

Backpropagation

Given the size of a typical deep learning model, it is an impossible task to guess the weights. A more reasonable approach is to estimate them by looking at the data. This where the concept of backpropagation [27] and gradient descent [28] are used. The training procedure of a deep learning pipeline can be broken down in to parts; Forward Pass and Backward Pass (Back propagation)

1. Forward Pass: In this step, the input is passed through the all layers. The input is multiplied by the weights of the first layers which is passed on to the next layer. This exchange of information between nodes keeps happening until it reaches the output layer where it is passed through an activation function to get the output in the desired format. Finally, the loss in calculated using this output value with respect to the actual value. This loss value is pass to the back propagation algorithm that updates the model weights.

2. Backward Pass (Backpropagation): The working principle of Backpropagation [27] is based on the gradient descent [28]. The ultimate goal of back propagation is to minimize the loss by bring the model predicted close to the actual output. Hence, this can be characterized as a minimization problem where the global minimum is needed. Hence the general idea is to calculate the gradient of the loss function with respect of the weights. In each iteration, the model weights are updated using the gradient of the loss function which guides the loss function towards the minimum. The equation used for weight update is $W_{t+1} = W_t - a(\Delta W)$ where a is the learning rate which tells the algorithm how fast or slowly it should move towards the minimum point and ΔW is the derivative of the loss function. This process of update is repeated several times until the model accuracy reaches an acceptable value.

2.2.3 AutoEncoders

An Autoencoder [29] [30] model could be used to learn the latent representation of the feature hence reducing the feature space to a much smaller size. It works by designing a model which reduces the dimension of features to a much smaller value and then by increasing the size of the reduced features to the original feature space. Such a model uses the input as the output of the model. The general idea is to train the weights of the down sampling and up sampling layers such that the bottleneck layer is as accurate as possible and to precisely represent the input data. If the model is trained properly then these latent features could use to represent the original feature space. An Autoencoder model can be used to encode the data by learning the compressed representation of the original data. The second part of the model is to train the weights to reconstruct the input from the encoded representation. While it is not possible to reconstruct the input with complete accuracy, the aim is to come as close to the input as possible. This is a dimensionality reduction [31] technique that is used to remove noisy features and only keep the relevant features. There are three main components of an Autoencoder which are also represented in figure 2.8.

1. Encoder: This is the first part of the model trains the weights of the associated layers to compress the data without losing information.
2. Bottleneck: This layers contains the encoded representation of the input data. This layer is often times referred to as the bottleneck of the model
3. Decoder: This second part of the model uses the compressed features from the bottleneck layer and tries to reconstructs the input.

2.3 Deep Neural Networks for Natural Language Processing

A standard deep learning model doesn't work well with a text related problem because the data involves a temporal component. For NLP related tasks, deep learning community has special models that are able to keep track of previous words while processing the current word in a sentence. Such models are know as Recurrent neural networks (RNN) [32] that

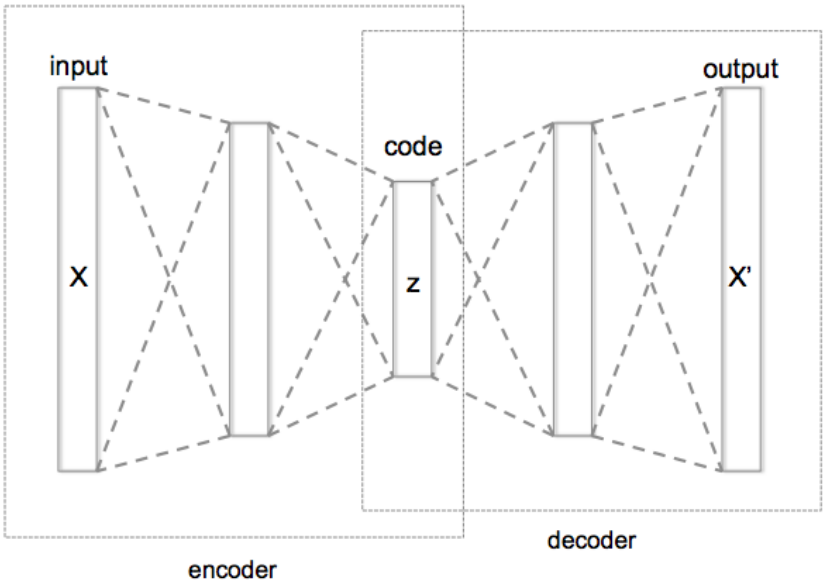


Figure 2.8: Abstract layers of an Autoencoder model where the model tries to reproduce the input through an intermediate dense representation.

have a memory compartment in them to keep track of the output from the previous word. Since RNNs can only keep track of the previous word, another variant of this model known as Long Short Term Memory [33] (LSTM) networks was introduced which was able to take care of long term dependency between words in sentence. RNNs and LSTMs would work very well for a next word prediction task where as language translation task would work well with a Transformer [15] model. A transformer model is built of an encoder and a decoder model which focuses on Sequence to Sequence tasks. While these models work well, they have their limitation such as the time, resources and data needed to train the model. But more importantly, all of these models are meant for supervised learning tasks which is not the case in this study. Since due to the last of labels, it is not possible to train a model, this study focuses on pre-trained model weights and word embedding, both of which are discussed below.

2.3.1 Word Embedding

Word embedding [34] [35] is a way of representing text before being passed as an input to the model. It is very different from previously used techniques such as Bag-of-word representation and TF-IDF [16] as it is able to capture the syntactic and semantic meaning of the text. They also allow to the text to be represented through a dense vector. The general idea of a word embedding is to have a multi-dimensional representation for each word allowing two words to be similar on some dimension and and dissimilar on others. One example would be to compare the embedding for the words 'pizza' and 'apple'. Since they are both edibles, they would values very close to each other in one dimension. However, as one is a fruit and other is fast food, the embedding values for another dimension would be far away from each other. A very simple example of a word embedding can be seen in figure 2.9. This specific example has an embedding size of 3 which represent the features [wing, engine, sky]. It can be seen that that goose, eagle and bee have a higher value along the first dimension because they have wings while helicopter, drone, and rocker do not so they have a value of 0. Similarly goose, eagle and bee have a 0 value along the second dimension as they do not have engines while helicopter, drone, and rocker do so they have higher values. In a typical word embedding model, each word can have up to 1024 dimensions. Essentially the embedding are generated for text so when a sentence of length 7 is passed through a model with embedding size 768, the resulting representation would be a matrix of 7 x 768. This means that there are 5376 (7*768) way of interpreting the sentence. A word embedding model is also trained like a deep learning model where the weights of the model are updated during the training phase. These models have a very intuitive architecture and training schemes, some of the models relevant to this study are discussed below:

Word2Vec

Word2Vec [2] [36] embedding model was first of it's kind and was trained using the word prediction technique. It followed two training schemes.

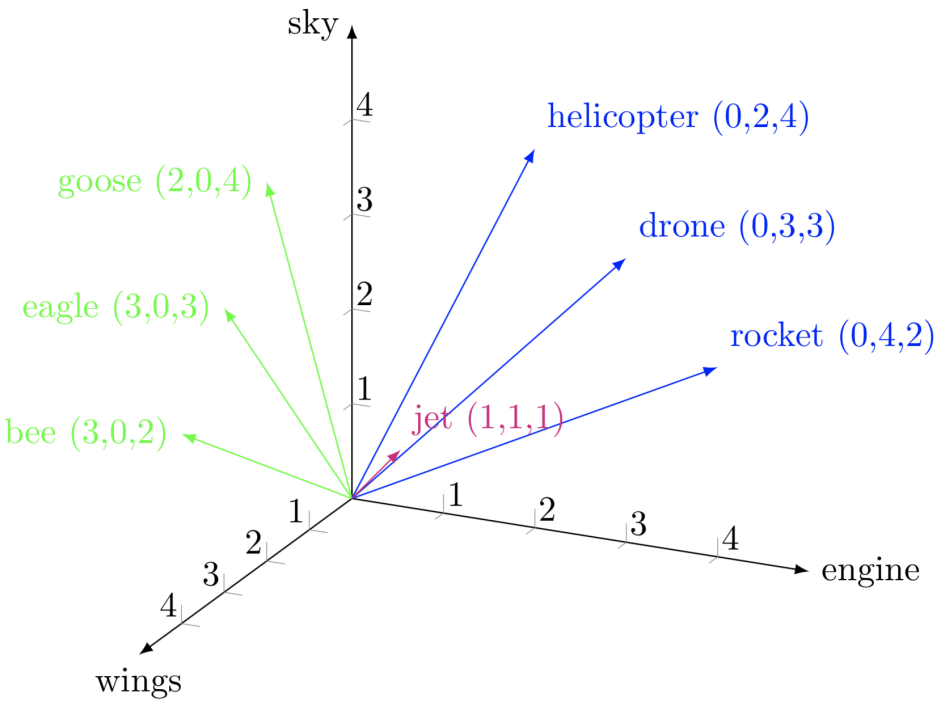


Figure 2.9: Visualization of a Word Embedding space with 3 dimensions.

1. CBOW model: This technique would iterate through a sentence and try to predict each word given the surround words. For a sentence 'Today is sunny', it would iterate through the sentence and train the model to predict 'Today' given the context words 'is' and 'sunny' and then try to predict 'is' given the words 'today' and 'sunny' and so on.

2. Skip-Gram mode:l This approach tries to predict the surrounding words given the context word. For the sample example sentence, given 'Today', model tries to predict the words 'is' and 'sunny'. Word2Vec model was first of it's kind but has several challenges such as the fact that the model architecture is not equipped to deal with unknown words, and has no concept of sub-words. The networks are not portable and cannot be used to initialize new architectures and it requires new embedding matrices to scale the model to a new language. Many state-of-the-art embedding model models such as BERT [3] and XLNET [5]have found solutions to these problems and are able to perform much better.

BERT

BERT [3] is a language model which uses two training schemes; Next Sentence Prediction in which the model tries the predict the next sentence given a first sentence and a word masking approach where the model randomly masks word in sentence and the model tries to predict them. BERT has two variants; . Both BERT and word2vec are unsupervised word representation methods but the main difference is that the embedding generated by BERT are

not static. They are generated for each word uniquely depending on the context. Generating word embedding though BERT requires the entire sentence to be passed through the model to get it's embedding vector, while for word2vec approach requires a lookup table. BERT base model which is built using 12 Transformer [15] models with embedding vector of size 768 and a BERT large that uses 24 transformers with embedding vector of size 1024. A Transformer model consists of an encoder and decoder part that focuses on sequence to sequence modeling. However, the interesting part is the self attention mechanism of the transformer model that tells the decoder block which words to focus on more in the input text when producing the output. The reason for BERT's success is the ease of scalibility and expandable nature. In general, BERT model is trained as a general model and can be fine-tuned for several down stream tasks. It also allows to get a very rich feature representation for the text using the pre-trained weights.

XLNET

XLNET [5] is a similar to BERT as it uses the same masking approach for training but it provides solutions to some of the challenges faced by BERT. Since BERT uses a masking approach for training by masking out words randomly in the sentence, the prediction for the masked out words might be correct on an individual level but often times doesn't make sense contextually for the text as a whole. For example; BERT model would randomly mask the words in a sentence as follows 'I went to [MASK] [MASK] and saw the [MASK] [MASK] [MASK]'. There are two ways the model can predict words for this sentence; 'I went to New York and saw the Empire State building' and 'I went to San Francisco and saw the Golden Gate bridge' are both correct. However if the model predicted 'I went to New York and saw the Golden Gate bridge' it would be contextually wrong but since BERT checks these predictions independent of each other, it would not consider it wrong. The main contribution of the XLNET model is to provide a solution to it by using an auto-regressive approach which checks these predictions in a sequential manner. However, in order to not have a unidirectional dependency, it first create a random permutation using which the words in the sentence are shuffled and then masks the words. This approach requires a lot of memory and hence makes it more expensive to train than BERT. XLNET was also trained on a dataset that is way larger than BERT which enables it to provide more contextual vector representations for the text. Another advantage that XLNET has over BERT is that BERT model cannot handle an input of more than 512 tokens where as XLNET has no such limit.

2.3.2 Transfer Learning

Transfer Learning is a term in Deep Learning which refers to using the pre-trained network weights. Since it is very resource consuming to train weights from scratch, it is usually a better practice to use the weights that have been generally trained previously and fine-tune it for a specific tasks. In simple words, the models weights are initialized with weights that were trained by someone else which makes the model converge way faster for a specific task. This is the main approach for word2vec [2] based method and also possible with BERT, to

initialize existing model with the word embedding and train it on another task. For BERT however, another possible approach is to fine-tune the BERT model itself on a new task, rather than initializing a different model. This is also a very good way of dealing with a dataset that is smaller in size. In the domain of NLP, transfer learning refers to using the word embedding weights to get a more contextual representation of the data. Since the state-of-the art models such as BERT [3] and XLNET [5] are trained on very large corpus, the vector representation is very accurate. The general idea is to use these weights to fine-tune a network for a more specific but that also requires labeled data. One the plus side, since these embedding are so strong, they work quite well even without the fine-tuning. So for a dataset that has no labels, it is possible to still get a meaningful feature representation to perform tasks such as clustering.

2.4 Clustering

In order to train a good Neural Network, the most important thing is labeled data which is quite challenge in a real world setting. This unavailability of good quality data gave rise to a branch of machine learning known as unsupervised learning. The algorithms belonging to this branch allow to get meaningful insight in the data without any labels. Clustering [37] [38] [39] is a very powerful technique that works by grouping similar data samples together. The output of a clustering algorithm can be seen in figure 2.10 where the data has been grouped into three clusters. Clustering a dataset has two main working

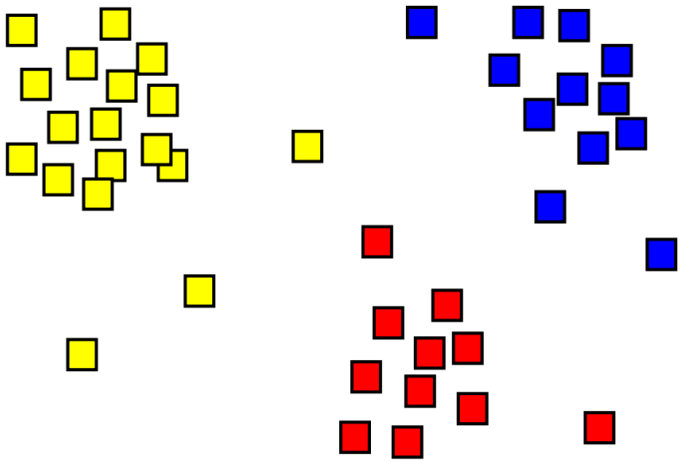


Figure 2.10: Output of a Clustering algorithm where data has been grouped into 3 clusters.

principles which are mentioned below:

1. The distance between data points within a cluster should be minimal.
2. The distance between the clusters should be maximized.

There are many clustering algorithms available but the one used in this study is K-means algorithm which is discussed in the section below.

2.4.1 K-means Clustering

One important hyper-parameter in clustering is 'k' which refers to the number of clusters. The general idea is to define 'k' clusters and assign one random data point to it as the initialization step which acts as the center point for that cluster. Then in the training phase, all data points are assigned to the cluster with whom the point has smallest distance. This process is repeated until the clusters no longer change or training has occurred for the specified number of iterations. To calculate the distance between data points, an appropriate distance measure has to be used. Most common choices of a distance function is the Euclidean distance which checks the feature-wise distance of two given data points. One possible approach of finding the clustering representative is to use the mean of all the data points belonging to that cluster. This techniques is known as k-means [40] clustering. The clustering obtained from a k-means algorithm varies quite drastically depending on the value of 'k' which refers to the number of clusters. This value for k depends on application, use-case or some prior knowledge about the data. In some cases, this value is known in advances. When clustering the customer review into positive and negative, it makes sense to use k=2, this is an example of application use-case. But, when a dataset contains text snippets from 4 different author then the value of k. However there are cases when no prior knowledge and information about use-case is available. In that cases, it is usually helpful to try various values for k visualize the results in order to find the value of k that best describes the dataset. The steps typically followed by a K-means algorithm to cluster the data:

1. Initialize k points randomly from the data. These points act as the centroids for each cluster.
2. For each data point, calculate the euclidean distance to all centriod and assign it to the cluster for which the data point has the shortest distance.
3. For each cluster, take the mean of all the data points assigned to it. This mean will serve as the centroid of this cluster for the next iteration
4. Keep repeating step 2 and step 3 until the data points belonging to a cluster no longer change or maximum number of iterations is exhausted

The above mentioned steps can be visualized in figure 2.11. Part 'a' shows a typical dataset with points not assigned to any cluster. Part 'b' shows two randomly selected data points for each cluster. Part 'c', 'd' and 'e' show glimpses of the convergence process that assigns data points to the nearest cluster. Part 'f' shows the final after the algorithm has converged.

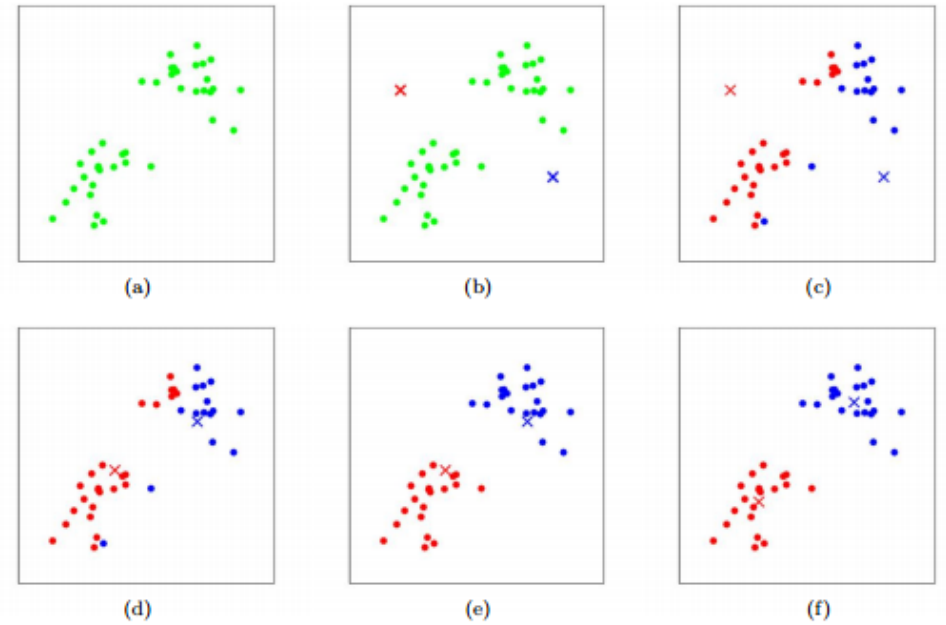


Figure 2.11: Convergence process of a k-means algorithm with k=2

3 Related Work

This chapter focuses on other techniques and research related to the tasks covered in this study. The first section of this chapter discusses some state-of-the-art techniques for Topic Modeling in general and evaluates them from a technical perspective. The second section focuses on works related to People Analytic which is a research area that uses Machine Learning techniques to analyze the data collected by Human Resources.

3.1 Topic Modeling

Topic Modeling [41] [42] is document clustering technique in NLP that groups similar documents together. It is an unsupervised techniques that also focuses on formulating a coherent theme that best describes each cluster through topic words. Topic Modeling can be used to analyze customer reviews, software logs, open-ended surveys, blog posts, employee objectives, etc. It is very helpful technique to provides automatic ways to analyzing huge collections of documents. It enables the users to saves a lot of time that would be otherwise spend on reading and analyzing all the documents. This is a very power approach as it doesn't need require any labeled data and extensive resource to train a model. It follows a very basic approach by learning how phrases and words co-occur in documents and uses that knowledge to group documents. In a simple scenario, it has one hyper-parameter which is the number of clusters (k) to which the documents should be grouped.

Latent Dirichlet Allocation [1] (LDA) developed by David M. Blei is a generative Probabilistic Model used for Topic Modeling. It is a very well formulated technique which is used by almost all researcher in the field as a benchmark for the evaluation of any new approach. This algorithm produces clusters based on the frequency of words in each document in the corpus and operates under the assumption that documents containing similar words are similar in nature. LDA uses a bag of word representation for text as input which simply refers to a vector that shows the count of each word in the text. For example, [3, 6, 7, 3, ..., 10] represents that the word with id 0 appears 3 times in the text and the word with id 2 appears 7 times. LDA model provides output in two parts; first is through a vector of length 'k' for each document that shows it's probability of belonging to each cluster and the second vector gives shows the probability of each topic word belonging to all cluster. For a text corpus of 100 documents and k=4, an LDA model would output document-to-cluster matrix of size 100 x 4 where for instance the values [0.1, 0.5, 0.3, 0.1] on 10_{th} row of the matrix would mean that the document with index 10 belongs to cluster 1 with 10% probability, cluster 2 with 50% probability, cluster 3 with 30% probability, and cluster 4 with 10% probability. Before converting the text to a bag-of-word representation, a dictionary is created out of the whole

corpus where each word is assigned an id. For each id, the model outputs a similar vector which tells the probability with which each word belongs to a cluster.

LDA2vec [43] by Moody is the first technique that combines classical approaches with deep learning to perform Topic Modeling. It works by combining an LDA [1] model with the skip-gram scheme of Word2Vec embedding. Instead of directly using the context words to predict the pivot word as done originally in word2vec training, lda2vec combines the document vector and word vector to form a context vector which is used to predict the context words. Moreover it also uses the concept of negative sampling where along with words that model should predict, it also passes the words that it should not predict during the training. When training a normal embedding model, for the pivot word 'German', the context word prediction would be 'French', 'English' or 'Italian' as the guesses are only limited to the local features learned during training. But the power of lda2vec lies in the fact that during training, it not only learns the low level embedding features, but also document related high level features which allows better guesses. Similar to LDA, the output of this model is also document weight vectors. The advantage of using lda2Vec over the original LDA comes into play when instead of human readable topics, machine-usable word level features are desired as output. lda2vec experimented with two datasets namely 'Twenty Newsgroups' and 'Hacker News Comments corpus'. In their study, it is shown that topics found by lda2vec correlates with the human evaluation. Furthermore, for the automatic evaluation of the results, an topic coherence score is used which tells how well-composed the clusters are. During their research, Twenty Newsgroups dataset yielded high coherence score whereas the model trained using Hacker News Comments dataset was able to find relevant topics within its community.

Similar to lda2vec [43], Embedded Topic Model (ETM) [44] by Dieng, Ruiz and Blei presents an approach that also combines LDA [1] with the word2vec embedding model. ETM works by learning a latent representation of the words similar to word embedding and also learning a latent representation for the documents in terms of topics. They experimented with 'New York Times' corpus and '20Newsgroups' corpus. They compared their results to an LDA model the neural variational document model (NVDM) [45]. They have used two type of evaluation for the experiments. First is by the qualitative measure which examines the word embedding generated by the models. Their study shows that the word embedding learned by ETM model are more diverse than the skip-gram and NVDM embedding. Second approach is the quantitative approach which measure the quality of topics by calculating an interpretability score. This score takes into account both the the topic similarity which shows how coherency of topics in a cluster and topic diversity which shows the uniqueness of the topics. For different vocabulary sizes, ETM seems to have higher interpretability score than LDA and NVDM for both datasets. Their technique presents an approach that is quite robust to stopwords and is able to produce better scores even when the stop words are not removed from the training data. This is not the case for other models.

3.2 People Analytic

People Analytic is an emerging research field hailing from several existing fields such as Data Science, Behavioral Science, Motivational psychology and predictive analytic. Given this digital age, many companies are trying to utilize the large amount of data at their disposal gain more insight into tasks such as estimating employee satisfaction, employee turn-over analysis, and employee recruitment etc. The main goal is to use Machine Learning tools to understand the employees and their needs on a personal level to design a more personalize management style for each employee. This study focuses on a task that tries to use state-of-the-art techniques to understand the yearly goals of employees by analyzing their objectives. Before diving into that, it is important to discuss some related work in this field and try to understand the approaches followed by them.

One task that has had a lot of attention from Machine Learning community is the automatic evaluation of employee performance. A lot of research is being done to find an automatic approach that analyzes employees performance and makes decision on appraisals and promotions. In a similar study, Sarker designed a hybrid framework [46] that uses Decision Tree and K- means Clustering approaches to predict employee performance. The dataset used in this study focuses on features such as written and verbal skills, sense of responsibility and punctuality among things. Another study [47] that proposes a model to predict employee performance was conducted by Nasr, Shaaban and Samir. They experimented with Naive Bayes, Support Vector Machine and Decision Tree to identify the factors that contributed towards employee productivity. The dataset used in their study was collected from the Ministry of Egyptian Civil Aviation on which they were able to achieve up to 79% accuracy.

Employee hiring is time extensive process that includes evaluation many candidates based on their experience, technical knowledge and social skills. For companies it is essential to hire someone brings value to the company and is reliable. Quite often, companies end up hiring employees who do not turn out to be a good fit for the company in the long run. To avoid that mistake, a lot of companies are inclining towards Deep Learning based solution that would help them pick the best candidate and to also reduce the cost of this process. The idea is to use an completely automatic pipeline that evaluates all candidates and picks the best profile. One approach for such a system is discussed by Faliagka [48] who presented a Machine Learning based algorithm for evaluation of candidate profiles for a recruitment process. Their approach uses some predefined features to evaluate the LinkedIn profiles of candidates and ranks them based on how suitable their profile is for the job. Their research uses classical approaches such as Linear Regression and Regression tree as a part of their rank learning algorithm.

Employee turnover prediction is another hot topic in the field of People Analytic. In human resource terminology, Employee turnover refer the act of replacing an employee with a new one. This could be caused by termination, resignation, death or retirement among other things but it often costs the companies a lot to replace an employee. Many companies are now turning towards machine learning approaches to predict an employee turnover so they better manage and plan accordingly. One approach was presented by Zhao in his research [49] to predict employee turnover within small, medium and large organizations. He experimented

with many variants of Decision trees [50], Support Vector Machine [51] models, Latent Semantic Analysis, Regression models, Clustering algorithms and Deep Learning models. The result of this study conclude that for a small dataset, results vary heavily depending on the nature of the dataset so it is recommended to try all models and see what words best. However if a bigger dataset is available then extreme gradient boosting [52] is recommended. These results were concluded by evaluating the statistical scores such as Accuracy, Precision, Recall, F1 Score, and Receiver operating characteristic (ROC) score.

4 System Architecture

The target group for this project within Merck Group includes employees with least amount of technical knowledge. To enable Merck Group to effectively use the configured Natural Language Processing models, it is important to have a system that is easily to understand and simple to use. The outcome of this study is a Flask [53] based application that provides a Graphical User Interface (GUI) for performing Topic Modeling on the employee objective dataset. The GUI provides an interface for working with the NLP engine and also provides various types of visualizations for analyzing the results. The main idea of this application is to get a holistic view of employee objective without having to read them. The application operates on a dataset provided by Merck containing the objectives of employees along with their anonymous ids. This application is meant for people in managerial positions and want to quickly get an insight into employee objectives. This section is divided into 2 sections. First is the front-end section which explains the navigation flow of the application and summarizes the available functionalities. The second section discusses the back-end and explains the NLP engine and flask end-points. A summary of the flow between the front-end and back-end of the system is discussed in the sections below and can also be visualized in figure 4.1 which shows how the results are pre-computed, stored to and read from the back-end.

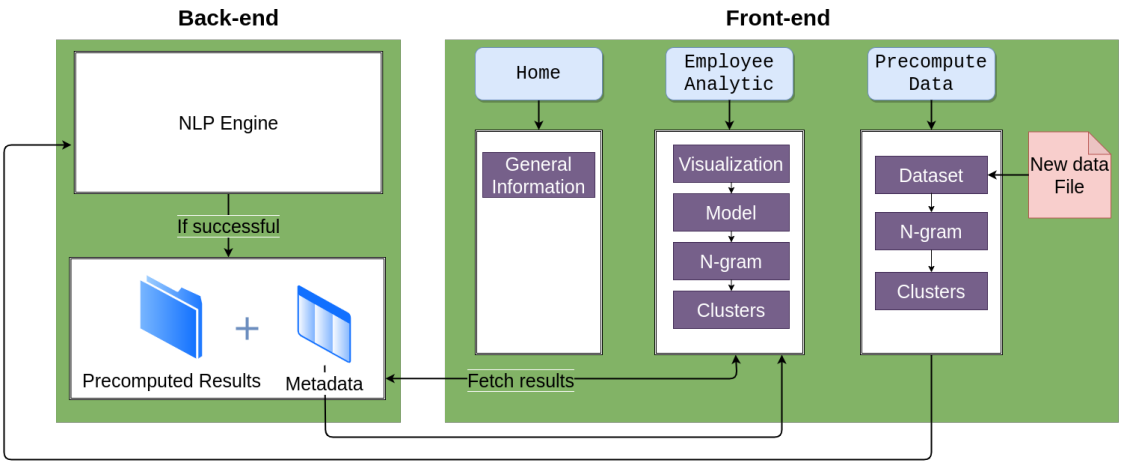


Figure 4.1: System Architecture of the Topic Modeling Framework that shows the interaction between the front-end and the back-end. 'Employee Analytic' tab reads meta-data from back-end before displaying options. It also fetches the requested results from back-end. 'Precompute Data' tab reads a new data file and generates results for all selected combination using the NLP engine and stores the result in back-end.

4.1 Front-end

The front-end of this application has three tabs; 'Home', 'Employee Analytic', and 'Precompute Data'. The functions and usage of each tab is discussed in a separate sub-section below:

4.1.1 Home

This is the start-up page and it contains some basic description about other tabs and explains how to navigate through them. A snapshot of the homepage can be seen in figure 4.2

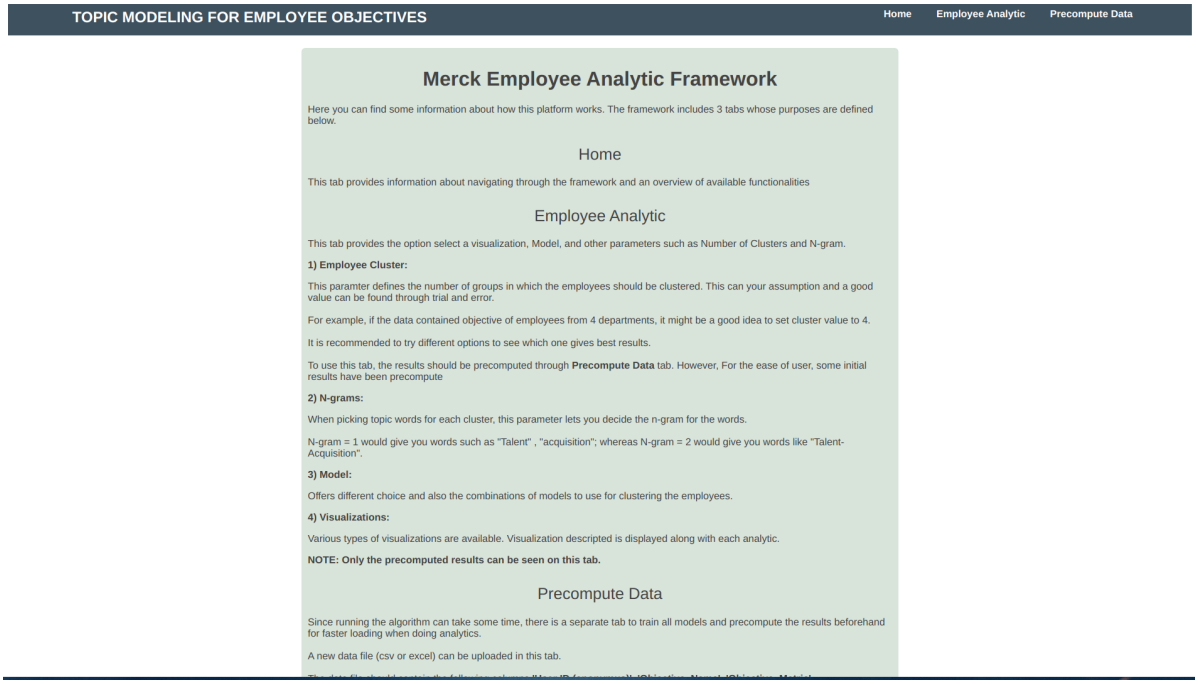


Figure 4.2: An illustration of the Home tab of the Topic Modeling Framework which provides an explanation about navigation and usability.

4.1.2 Employee Analytic

This tab is used to configure different parameters for clustering. Results for all of the configurations need to been precomputed beforehand in order to provide a smooth user experience. More details on it are discussed on the section 4.1.3. Furthermore, as shown in figure 4.3, there are 4 important configurations to be selected to visualize the results including type of visualization, type of Model, Employee Clusters (k), and choice of N-gram. More details on this configurations are discussed in the sub-sections below:

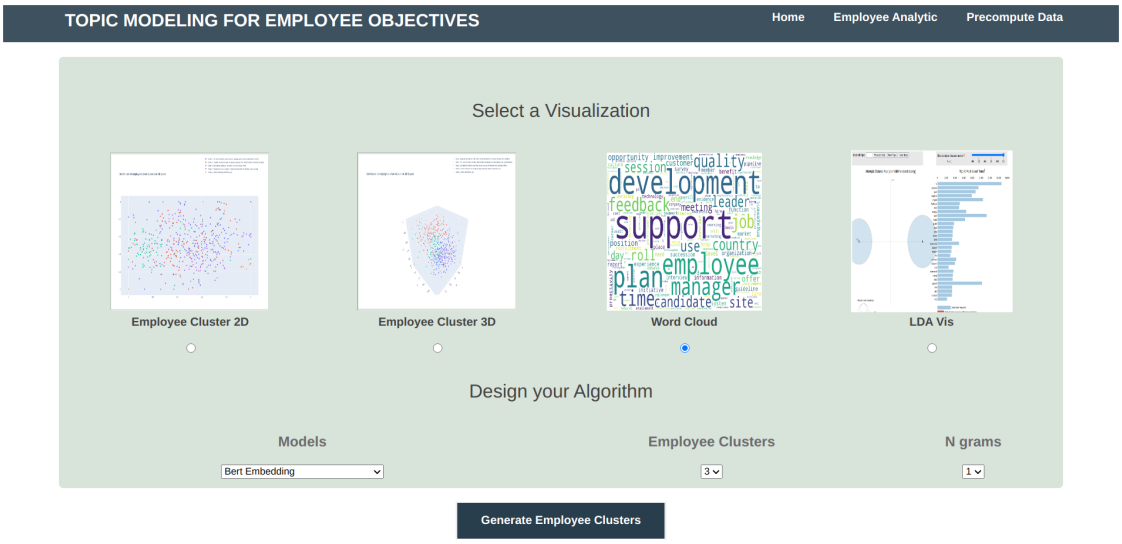


Figure 4.3: An illustration of the Employee Analytic tab of the Topic Modeling Framework which allows to select the type of visualization, model, employee clusters and N-gram for results visualization.

Visualization

This framework includes four types of visualization to analyze the results from the NLP engine and are mentioned below:

- A 'Word Cloud' refers to an image which contains main topic words and the font size of each word represents it's respective relevance. The number of word cloud images depend on the configuration of Employee Clusters. In this framework, each word cloud represents the main topics from the objectives of employee in a cluster. A word cloud is a very basic but powerful visualization which helps formulate the theme of each cluster. This visualization shows the results of clustering and topic word retrieval combined. A sample word cloud output with two Employee clusters is shown in figure 4.5.

- 'Employee Cluster 2D' is a type of visualization that displays employee clusters in two dimensional space. However, this type of visualization is meant for the clustering part of Topic Modeling as it only displays the clusters with little emphasis on selected topic words. A snapshot of this visualization can be seen in 4.6 where employees are clustered into five groups. In the graph, each dot represents an employee and it placement in the space with respect to their objective while the color represents the cluster they are assigned to. On hover, it is possible to get see the employee id and a glimpse of their objective. The graph legend on top right shows relevant topic words for each cluster.

- 'Employee Cluster 3D' represents the employees in a three dimensional space. This visualization is very similar to the two dimensional representation except for the fact that

it is able to retain more features. A snapshot of employees plot in a three dimensional space can be seen in figure 4.7.

- 'LDA vis' is a visualization created by pyLDAvis [54] which is a python library that provides tools to visualize an LDA [1] Model. The visualization provided by pyLDAvis can be seen in figure 4.8. This visualization shows the probabilities predicted by the LDA for document-cluster and topic-cluster probabilities. The circles on the left for example show the size of each cluster depending on how objectives belong to it. On the right, for each cluster it shows the top words and the number of times each word appears in it. This visualization is only available when using an LDA model. so when visualization is selected, the dropdown menu for model selection is disabled as shown in figure 4.4.

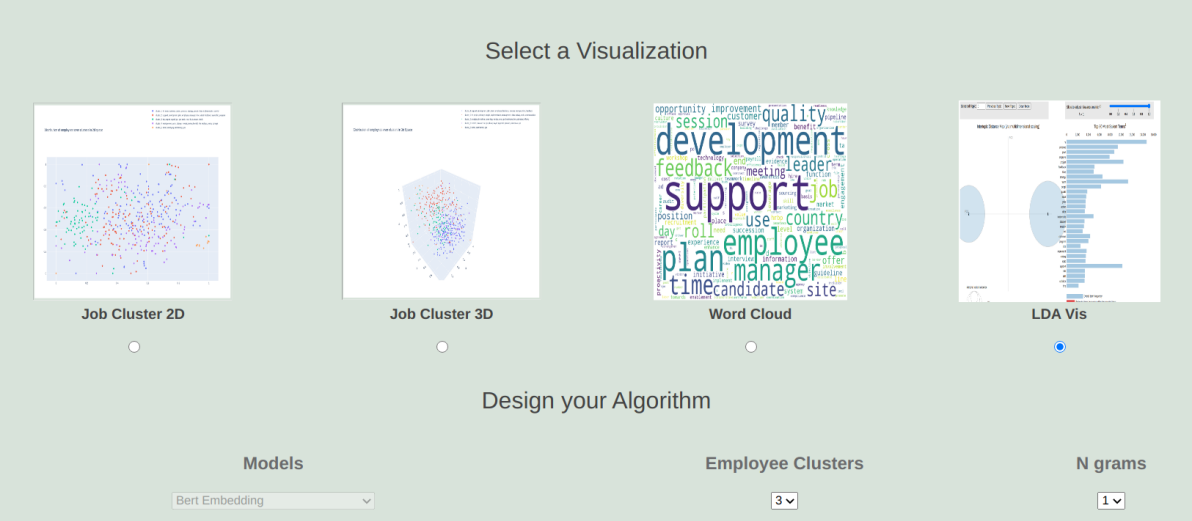


Figure 4.4: System behavior when LDA vis is selected as visualization. The Models dropdown menu is disabled as 'LDA vis' only works with LDA model so a model selection is not possible. The system automatically selects LDA mode.

Employee Cluster

This parameter represents the number of clusters the algorithm should be group the employee into. This is an assumption and usually the correct value can be found by trial and error. For Instance: If the data had objectives of employees from 4 departments, a good assumption would be that employee cluster is 4. The dropdown menu for this parameter shows only those values whose results have been precomputed through 'Precompute Data' tab.

Model

This configuration offers different choice of models to use for clustering the employees. The choice of models includes two type of word embedding, the LDA model, and also the

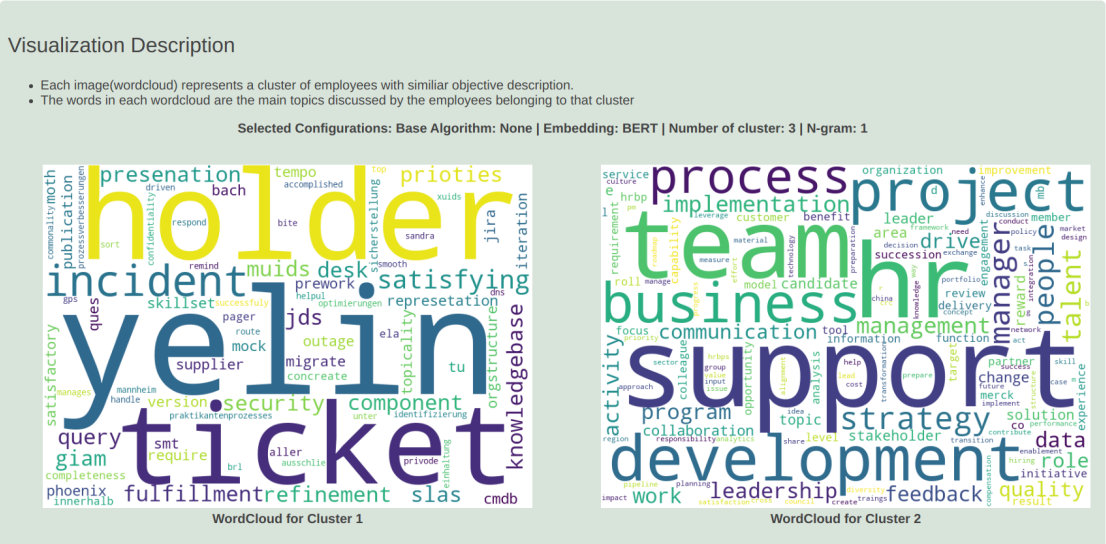


Figure 4.5: A sample on Word Cloud Visualization for 2 clusters on Employee Objective dataset. The plot shows a summary a selected configurations and and visualization details.

combination of word embedding with an LDA model. Since, the LDA model works with N-gram = 1 so whenever it is selected, the dropdown menu for N-gram is disabled as shown in fig 4.9.

N-gram

When picking topic words for each cluster, this parameter lets you decide the n-gram for the topic words. N-gram = 1 would give you words such as "Talent" , "acquisition" whereas N-gram = 2 would give you words like "Talent-Acquisition". The dropdown menu for this parameter shows only those values whose results have been precomputed through 'Precompute Data' tab.

4.1.3 Precompute Data

This tab allows to precompute results for new data. Since it takes times for the NLP engine to compute results on run-time, the results for all combination are precomputed and saved. To provide a smooth user experience, the results are simply loaded and displayed instead of computing everything on the fly. As seen in figure 4.10, the results can be procomputed by uploading a file (csv or excel) with new data. A successful precompute overwrites the data from the previous precompute. The dialog also allows to select the values of n-gram and employee clusters for which the results are precomputed. The Employee Analytic tab only shows the values of N-gram and employee cluster for which the results have been precomputed here.

Figure 4.6: An illustration of Employees grouped into 6 Cluster in 2D space using Employee Objective dataset.The plot shows a summary a selected configurations and and visualization details.

Progress of the precompute operation can be tracked using the progress bar as shown in figure 4.11

4.2 Back-end

The Back-end of the Topic Modeling Framework uses Flask [53] api to manage end-points and an NLP engine as the backbone. Some relevant end-points along with their roles are mentioned below:

4.2.1 Home

This is the startup page for the framework which contains some html code on the back-end to display hard-coded text as seen in figure 4.2.

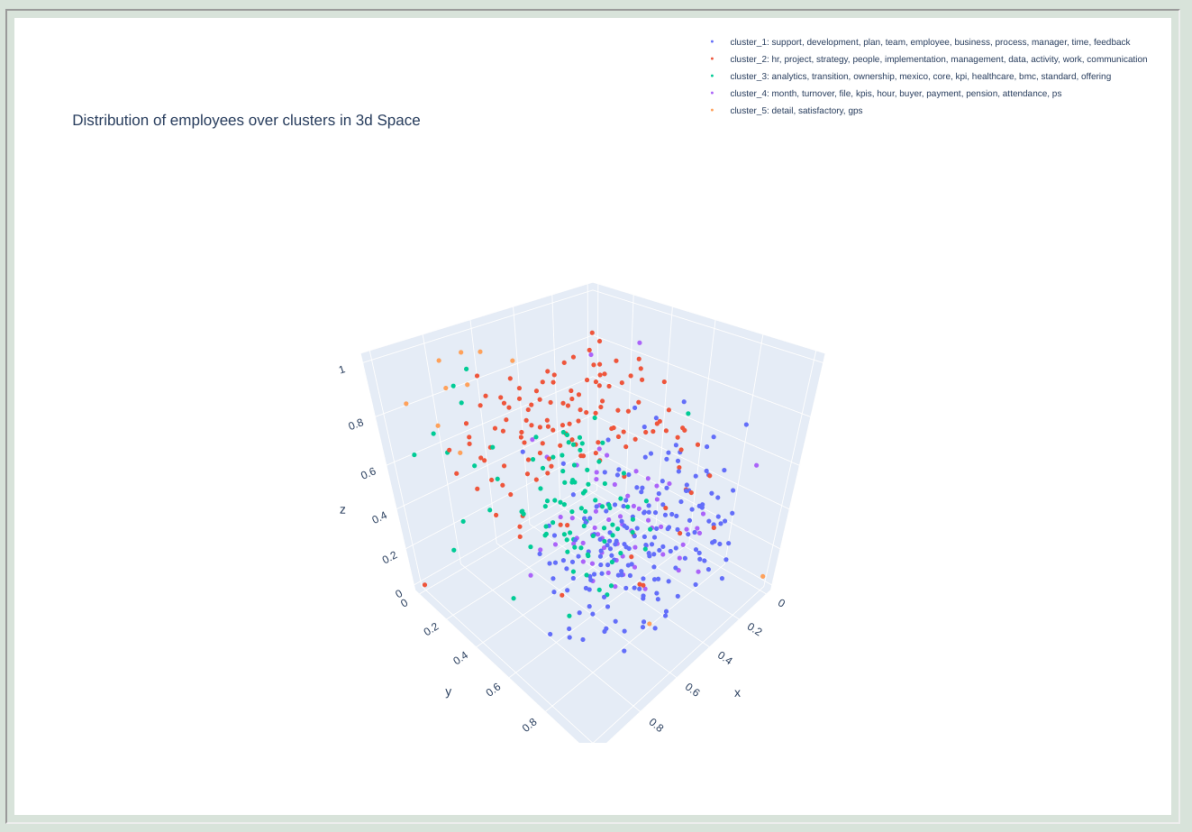


Figure 4.7: An illustration of Employees grouped into 6 Cluster in 3D space using Employee Objective dataset.

4.2.2 Employee Analytic

When a user click on the 'Employee Analytic' tab, the analytic end-point shows the results or an error message after following these steps:

1. Reads the meta-data json file from the previously precomputed results as shown in figure 4.1. This meta-data file has information about the values of N-gram and Employee Cluster for which the results are available. On the front-end, it then only displays the values for users for which the results are available.

2. The loaded interface allows user to select the configurations for which they want to see the results.

3. When 'Generate Employee Clusters' buttons is clicked, it connects flask to the 'result' point and sends all the selected configuration along with the api call.

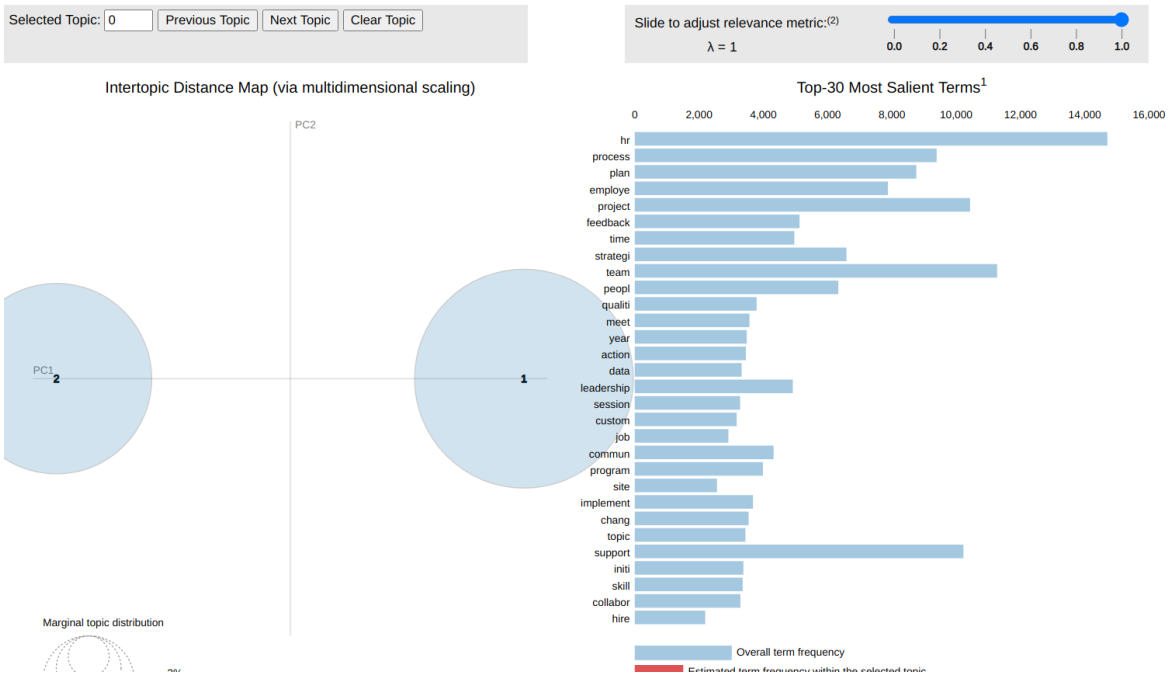


Figure 4.8: An illustration of 'LDA vis' Visualization who gives insight into the probability scores of an LDA model.

Results

When this end point is triggered, it generates the path to the output folder where the request results should be present and does the following:

1. It checks if the output folder exist.
2. If the output folder is missing, it simply shows an error message that these results are missing and need to be precomputed.
3. If the folder is available, it reads the html files or images and displays them along with some visualization description.

4.2.3 Precompute Data

User is connected to this end point when the 'Precompute Data' tab is select. This endpoint provides an interface to upload a new file and precompute results on for the selected configurations of N-gram and Employee Clusters which can be seen in figure 4.10. When the 'Precompute on selected configurations' button is clicked it does the following:

1. Checks that a valid file is selected, at least one value for N-gram and Employee Cluster each have been selected.

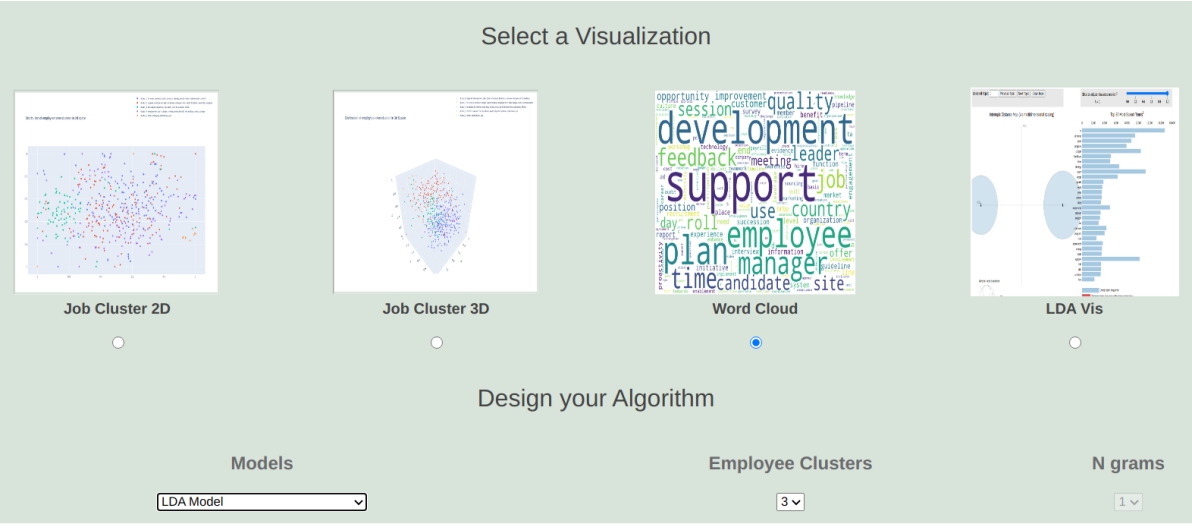


Figure 4.9: System behavior when LDA model is selected. Since this model only works with 1-gram, the dropdown menu for N-gram is disabled.

2. If the above mentioned checks are full filled, it proceeds with precompute and directs to the progress end-point along with the selected configurations. The progress end-point is discusses in section 4.2.3.

3. otherwise it redirects to the error page end-point.

Progress

This page shows a progress bar which tracks the progress of the precompute operation. If for some reason, there is an exception during the precompute then it also show an error immediately. There is also a 'Cancel Precompute' button that aborts the whole function as shown in figure 4.11. The summary of the progress end-point is discussed below:

1. This end-point first reads the file and pre-processes it using the steps mentioned in section 5.1.

2. It creates a temporary folder to store all results.

3. Writes the meta-data to a '.json' file to be read by the Employee Analytic tab.

4. For the select configurations from precompute end-point, all visualization and model choices, it iterates over it

5. For each combination, it creates an output folder and a 'Topic Model' Object which does features selection as mentioned in section 5.2 using the selected Model. It also performs clustering and get topic words as mentioned in section 5.3.1 and 5.3.2 respectively.

Figure 4.10: Pre-compute Tab of Topic Modeling Framework. It allows user to select a new data and also select the configurations for which results should be precomputed.

6. It saves them results in either '.html' or '.png' depending on visualization in the generated folder.
7. If results for all combination are precompute successfully, it replace the previously precomputed results with the new one.

Error Page

This is simple html page that displays an error message depending on the exception.

- If the user forgot to upload a file, then it displays a message accordingly as shown in figure 4.13.
- If the user did not select a value for N-grams or Employee Clusters then it displays a message accordingly as shown in figure 4.12.

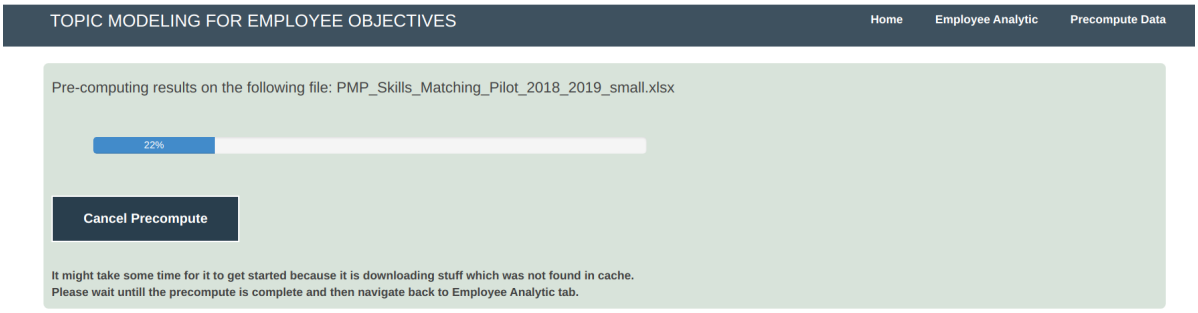


Figure 4.11: A snapshot of progress bar to track the precompute operation. 'Cancel Precompute' allows to cancel the operation.

Figure 4.12: A snapshot of error message when user forgot to select configurations for precompute.

Figure 4.13: A snapshot of error message when user forgot to select a file for precompute.

5 Methodology

In this study, the Topic Modeling pipeline is distributed into fives steps. First is the pre-processing of data before being passed to the model, second is the feature selection step for clustering, third is clustering the documents into groups, forth is the topic retrieval step which finds topics words for each cluster, fifth and last is a post-processing steps which removes redundant topic words from clusters. An abstract concept of the methodology is presented in figure 5.1 which depicts the whole pipeline as well as the types of available feature spaces.

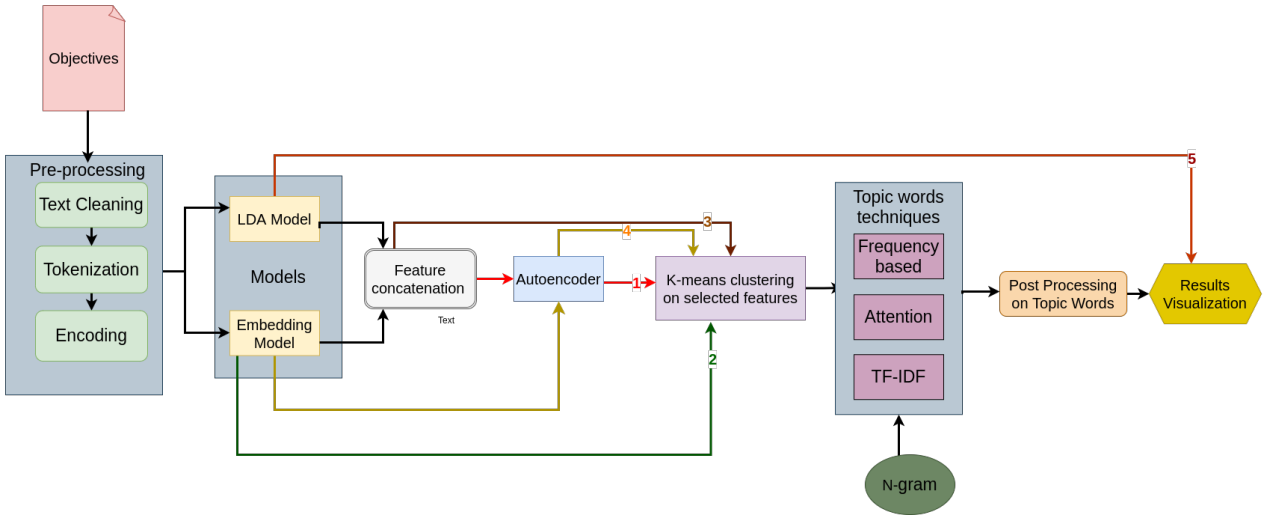


Figure 5.1: A Block diagram of the methodology used in this study. An objective document is passed through a Pre-processing step first, then one of five type of feature spaces is selected which is used of clustering. Next, one of the three topic word retrieval technique is selected to get the top topic words. Last step is post processing to remove redundant topics from clusters.

This chapter is divided into five sections. First section pre-processing describes the steps used to prepare the data for cluster. Second section discusses the choice of features for clustering while third section explains the clustering techniques used in this study and also discusses the importance of number of clusters (k). Fourth section discusses the techniques used for finding topic words for each cluster. Fifth and the last section is about the post processing techniques used to get more coherent cluster themes.

5.1 Pre-processing

Pre-processing is an important step in any Natural Language Processing pipeline as it converts to textual and unstructured data into a form that is understood by the model. This study focuses on two types of models, LDA [1] model and Embedding models. Both of these models require the data to be pre-processed differently. An LDA model requires input data to be in bag-of-word representation. Such a representation disregards any grammatical form and order of words in a sentence. Hence, only focusing on the word count. The beauty of word embedding model is that they don't this extensive manual pre-processing as an LDA Model. Most Embedding models come with their own tokenizer which takes care of all model specific pre-processing. Moreover, they have built-in dictionaries which contains the vocabularies used for model training. These models have techniques of dealing with unknown words such as breaking it into sub words that exist in the model vocabulary. Before passing the data to an embedding model. While the pre-processing steps are necessary for a word embedding models from hugging face library, the sentence embedding models such as Sentence Bert and Sentence Roberta from Sentence Transformer library do not require any pre-processing at all and text is passed directly to the model. However, some general pre-processing steps include text cleaning, tokenization, lemmatization, and encoding into a numeric form.

5.1.1 Text Cleaning

This simple step cleans the text by removing the words and characters which add no value to the features. This noise removal step procedure is essential for all models as it makes the feature more meaningful and improves the quality of results. This text cleaning processes can also be regarded as a smoothing process which removes typos and redundancies.

5.1.2 Tokenization

Tokenization means breaking a sentence into a list of words (features) which can be fed to the model. For example: 'My name is Adam' would be tokenized as ['My', 'name', 'is', 'Adam']. There are many existing libraries that perform this tokenization. For Word Embedding Model, tokenization step is a different as it requires a model specific tokenizer from huggingface [55] library. The reason for using a model specific tokenizer is that some models like BERT and XLNET are essentially trained on a sentencepiece [56] vocabulary which works by breaking down unknown words into smaller sub-words and treat them as individual words. Since this is the vocabulary that the model was originally trained on, it is required to convert the text into the words of the same vocabulary.

5.1.3 Encoding

Encoding refers to replacing the tokenized words with their respective ids. This is usually done first creating a dictionary that contains all the unique words from the corpus and assigning them an id. It possible to create a dictionary and encode all tokenized words using

the gensim [57] library. However, the Embedding Models come with their own dictionary containing the vocabulary on which they were trained. Similar to the tokenization step, the encoding for an embedding model is done through the model specific tokenizer. Since it is faster to pass data points to an embedding models in batches, all the encoding vectors must be padded with a model specific pad id to ensure same length.

5.2 Feature selection

The results of a clustering algorithm heavily depend on the features used for grouping the data points. This study focused especially on acquiring meaningful and context rich features by using state-of-the-art techniques. Two main sources of features extraction include learning embedding vectors from different type of word embedding model and using the document-to-cluster probability vector from an LDA model. This study focuses on evaluating the results from both these sources as well as combining these features to see if the tandem can give better results. This study also presents the concept of using an Auto Encoder to learn the latent representation of the three types of features mentioned above and use it for clustering. The methodology used for obtaining these features is discussed in the sub-section below:

5.2.1 Latent Dirichlet Allocation Features

Latent Dirichlet Allocation (LDA) [1] is a probabilistic model which is used especially for Topic Model tasks. As already discussed in detail in the section 3.1, this model outputs a vector that provides the probability of each document belonging to a cluster. In this study, the results from an LDA model are used as a baseline to compare results. But it has another purpose, that is combining the feature vector from an LDA model with the feature vector from the embedding model to get both local and global representation of the text.

5.2.2 Word Embedding Features

Topic Modeling is an unsupervised task, which automatically entails that training or fine-tuning a model on the final layers is almost impossible. However, since the embedding models are trained extensively over a large dataset, it is possible to use their weights to get context-driven features of the text to be used for clustering [58]. Word Embedding models provided a token based representation of the input sentence. For a model with embedding size 'm', it returns a vector for length 'm' for each token. If a sentence has 'l' tokens (words), then the model would output a matrix of size [l x m]. Similarly, if a batch of 'N' sentences with 'l' tokens each is passed to a model with embedding size 'm', the output dimensions would be [N x l x m].

Sentence Embedding: For a problem such as text clustering, the token level embedding representation doesn't do much as this study deals with learning the feature of the employee

objectives as a whole. There are multiple ways of deriving a sentence level representation which is a vector of length 'k' where k is the embedding size of the model. A popular approach involves taking the mean over all tokens to get a sentence level representation. This approach is also followed in this study and more details on it are mentioned in the experiments chapter. While this approach of taking mean over all tokens works for many models such as XLNET [5], XLM [59] and Electra [60], there are some models like BERT [3] and it's variants which are trained to provide a sentence level representation of the text as well. BERT Model is originally trained for two tasks; next sentence prediction and text classification. For this purpose, they have provided a [CLS] token which provides the representation to be used for classification tasks and a [SEP] token which represents the end of a sentence. For Bert, and all it's variant models such as RoBERTa [61], and DistilBERT [62] are trained with a [CLS] token which provides the sentence level representation and can be used for clustering.

Another variant of Embedding Models are the sentence embedding where the models have been adapted to output a sentence level representation which captures the context of the sentence as a whole rather than of individual tokens. One example of such a model is Sentence Bert [4] which outputs the embedding vectors much faster than the normal BERT. It uses a Siamese and triplet network structures to get a more meaningful sentence representation which are more effective for a clustering task.

5.2.3 LDA Features + Embedding Features

One of research questions in this study is to evaluate if using the embedding feature vector along with the LDA [1] model features could provide better results than using the feature spaces separately. The idea behind this theory is that while LDA model features provide high level information, embedding feature vector could provide the low level insight to the data by capturing the context across multiple dimensions. Combining these two set of features could potentially lead to much better clustering. The general idea to concatenate these two feature vectors next to each other and use them as a whole for clustering.

5.2.4 Auto Encoder

Word Embedding models have a huge feature space and not all features are necessary for the clustering task. The purpose of using an Autoencoder is to extract only the relevant features and use them for clustering. This study uses an Autoencoder with five layers in both encoder and decoder each. The dimensions of each layer are dynamic and depend on the size of the input. For the encoder, the first layer reduces the size to 80% of the input dimension, the second layer reduces it to 60% of the original input dimension where as third and fourth layers reduces it to 40% and 20% respectively. The final layer brings to the either a latent dimension of 32 features or 10% of the input dimension if the previous layer had features less than 32 features. The decoder works in a similar way by slowly increasing the feature size to 20%, 40%, 60%, 80% and then finally to same dimension as the input. Once the model has been trained, the original features are passed through the encoder to obtain a latent representation which is then used for clustering. This dimensionality reduction approach

is used on features from the embedding model and also used when the features from LDA model and Embedding model are concatenated together. Clustering is done after performing dimensionality reduction and also on the original feature space and results are compared for all four variants to the results from LDA model.

5.3 Topic Modeling

5.3.1 K- means Clustering

Clustering is an unsupervised Machine Learning approach can be used for clustering employees based on objective. Once the objectives have been pre-processed and converted in to a feature space containing numbers, the next step is to cluster the employees based on these features. For clustering, a K-means clustering algorithm is used which groups the employees on the basis of similarity between their respective objectives. The technical details related to K-means clustering are discussed in the background chapter in section 2.4.1. In this study, 'k' is a hyper parameter that refers to the number of clusters into which the employees should be grouped. Since the employee objective dataset from Merck Group is very unstructured and there is no prior knowledge available about the possible groups, the parameter k is set as configurable. In this study, clusters were generated for different values of k whose results are discussed in the Results chapter.

5.3.2 Topic Words

Simple clustering of text is not very as intuitive because there are no labels available for clusters. In a topic modeling approach, the next step is to find topics related to each cluster. This can be broken down into two aspects, first is the techniques used to find the topics and second is the N-gram which tells how many words should be used to describe a topic. Both of these approaches are discussed in more details below.

Techniques:

There are many ways of finding topics such as picking the words that occur the most in the cluster, or using a TF-IDF approach for finding unique word as, or using the attention vectors from the word embedding models to find important keyboards. There are a few restrictions as not all techniques can be used with all models. A summary of these restrictions are provided in table 5.1 which shows what features space could be used for which configurations of topic word techniques. For example, the self Attention mechanism does not work with Sentence transformers as they do not output the attention vectors. LDA [1] model is not included in the list as it has its own mechanism for finding topic words which is used as the baseline.

1. Transformer Attention: The Word Embedding models used in this study contain multiple transformers [15] as building blocks. The total number of transformer blocks in an Embedding model varies. BERT [3] (base) uses 12 transformer blocks, BERT large uses 24 transformers

Table 5.1: Possible combinations of features with topic word techniques. Transformer attention cannot work with Sentence Embedding models as they don't contain attention vectors. LDA model is not included as it has own topic word mechanism.

Feature Space	Frequency-based	TF-IDF	Transformer Attention
Sentence Embedding	✓	✓	x
Word Embedding	✓	✓	✓
LDA + Sentence Embedding	✓	✓	x
LDA + Word Embedding	✓	✓	✓

blocks while XLNET uses an adaption of a Transformer block known as Transformer XL. These transformer blocks have a feed forward Layer and an additional Attention layer which depicts how much attention should one word pay all other words when producing outputs. A standard transformer used in BERT model has 12 or 24 transformer with 12 attention heads each. Each head in a transformer layer is responsible for specific textual feature just like each layer in a deep neural network learns a feature of the data. The role of each head in transformer layer is unknown but collectively they aid the feed forward layer in producing more accurate results. For the sentence 'The monkey ate that banana because it was too hungry', what does the word 'it' refers to? For a human, it might be quite simple to understand this but it is not easy for the model. As the model iterates through each word of the sentence, the attention mechanism evaluate how much weight it should allocate to the words before it. This attention weights are usually represented with a number but to a visual representation of it can be seen in figure 5.2 which depicts how a transformer block deals with long term dependencies in text. In the figure, the thickness of the connection between the words shows how much attention should the word 'it' pay to all the words before it. Intuitively, the attention values could be used to determine the topic words in the cluster.

2. TF-IDF: Term Frequency - Inverse Document Frequency (TF-IDF) as discussed in section 2.1.2 is a well-known approach used to find important keywords in a text. It works by assigning higher weight to words that have more frequency within it's own document and lower frequency in other documents.

3. Frequency-based: This technique is based on the assumption that the words that appear in a cluster more often have more importance. Hence, the words with higher frequency can be used as the representatives of the cluster.

N-grams

It is possible to find phrases that define the clusters rather than just single words. This is a popular approach in NLP which is referred to as N-grams. 'N' is a configurable hyper parameter in this study. A 1-gram approach focuses on finding single words such

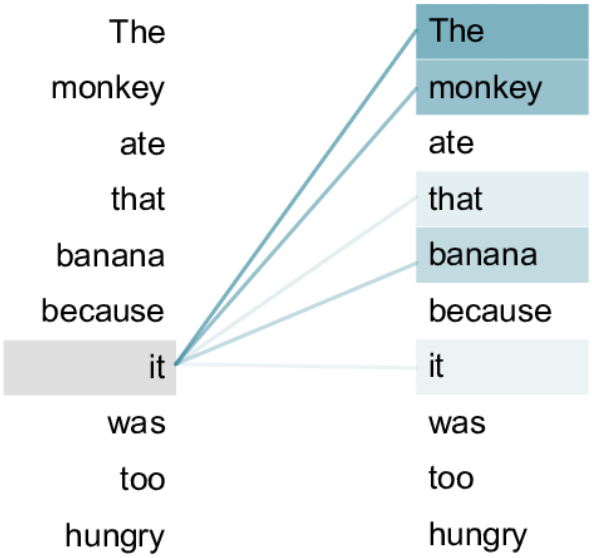


Figure 5.2: An illustration of the self Attention mechanism of a Transformer block where the thickness of line depicts the strength of attention the word 'it' should pay to all the words before it.

as 'Development', 'Leadership' and 'Strategy' as topic words. Whereas a 2-gram approach focuses on finding phrases containing 2 words as topics such as 'meeting-expectation', 'action-plan' and 'talent-development'. Similarly a 3-gram approach tries to find phrases such as 'application-approval-process', 'project-milestone-result', and 'manage-project-activity'.

5.4 Post Processing

Given the nature of the dataset, It is expected for the cluster to have some common topic words such as 'goals' and 'milestones'. These common topic words would result in less precise theme for the cluster. To avoid these overlapping topics in the clusters, a post processing step is used to make sure that the topics within each cluster are distinct and provide a unique theme. Two schemes are introduced in this study for post processing the topic words which are discussed below:

5.4.1 Hard Policy

This policy deletes the redundant topic words from all cluster. For example, if a topic words exists in more than one cluster, then that would not be used as topic word for any cluster.

5.4.2 Soft Policy

This policy operates on a frequency of words and keeps words in cluster with maximum frequency. If more than one cluster contained a topic word, it would keep the topic word in

the cluster which had the highest frequency count for it and deleted from all other cluster. This approach allows the topic words to belong to the clusters where they are most relevant as depicted by the frequency count.

6 Datasets

The Employee Objective dataset used in this study was provided by Merck Group. Since the Employee objective dataset is noisy and smaller in size, an additional Job description dataset is used to run experiments and to find the best configurations for model parameters. While the first dataset has privacy constrains, the second dataset is open source and is downloaded from Kaggle. Both of these datasets are discussed in details below:

6.1 Employee Objective dataset

The employees at Merck Group are required to write their goals and plans for the coming months in the form of objectives. Merck Group compiled and provided a dataset containing the objectives of some employees. This columns and a sample value contained in this dataset are shown in table 6.1. For privacy reasons, the text in objective is redacted from several place as it is a private dataset.

6.1.1 Dataset Analysis

This dataset is however, very sparse and unstructured. For example the fields 'Objective Comment' and 'Objective Description' are missing for most objectives. Secondly some of the columns such as 'Global Key', 'Functional Area', 'Form Template Name' and 'Location' are not relevant to topic modeling task in this study. For the above mentioned reason, a more simpler version of the dataset with only three columns as shown in table 6.2 is used in this study. The three columns 'User ID', 'Objective Name' and 'Objective Metric' contains the information which is relevant for clustering employee based on their objectives.

The dataset was analyzed using many features such as the redundancy, length and language of the objectives. Some findings from the analysis are discusses in the sub-sections below:

Redundancy in Objectives

One issue with this dataset is that it has a lot redundant entries. Objective texts were copy pasted to create new entries which resulted in a dataset of approximately 33,000 objectives. However, after removing all redundancies, the dataset had only approximately 4078 unique objectives left.

Table 6.1: A censored sample data point from the Employee Objective Dataset with all columns. Some of the text is redacted to ensure the privacy of data as it is a private dataset.

Column Name	Sample Data
User ID (anonymous)	****0000
Global Key	HR
Functional Area	Human Resources
Location	Temecula
Objective Name	Operational Excellence
Objective Description	None
Objective Comment	I appreciate that Kathy has been continuing to work * * * * * * is a vital part of our support to our * * * * * * * * * * * * * * * * * supports * information which is very much appreciated. I also * * * * * * * * * , TOA to Sick Time and recently * * * * project among *. Thank You, Kathy! * * * * * November 2019.
Objective Metric	Support * * * * * and other activities focused around * and * for *. Support the * * *, includes * *, * * and other * along the way. Also expected is a regular * * * * * accurate * is recorded in * * * * * * * and is reflected in * * or selected tool. * * * * * * initiatives * * * *. * and * * * * the * of key insights for a further deep dive utilizing *. Continue to * * knowledge of * * * * * * cluster. * * * on a regular basis. Support efforts * from the * * * * * * management and the * for * * Acknowledgement of same.
Form Template Name	2019 Performance Management

Table 6.2: Merck employee objective dataset with selected columns to be used in study.

Column Name	Column Description
User ID (anonymous)	This column contains the anonymous employee ids
Objective_Name	This column contains the name of the Objective
Objective_Metric	This column contains the objective description

Statistical Analysis on Objective length

A statistical analysis on the 4078 unique objectives is illustrated in figure 6.1 which shows the summary of number of words in each objectives. The shortest length of an objective is 0, which means there are some objective with no text at all and the longest length of an objective is 492 words. Moreover, there are very few objectives that have length of greater than 200 words. Statistically speaking, 99% of the objectives have length equal to or less than 219 words. For this reason, all the objectives with length greater than the maximum size among the 99% of the objectives were truncated to 219 words. The advantage of this truncation is that language model require a fixed length for all texts. Hence, the standard practice is to pad all the sentences shorter than the maximum length with '0' or a fixed id to ensure fixed size. If the maximum length is set to be 492 words then it increases the memory requirement drastically while not not adding equivalent value to the feature space. Truncating the text to the maximum length within 90% of the objective allows to still maximal features while keeping the memory requirement reasonable.

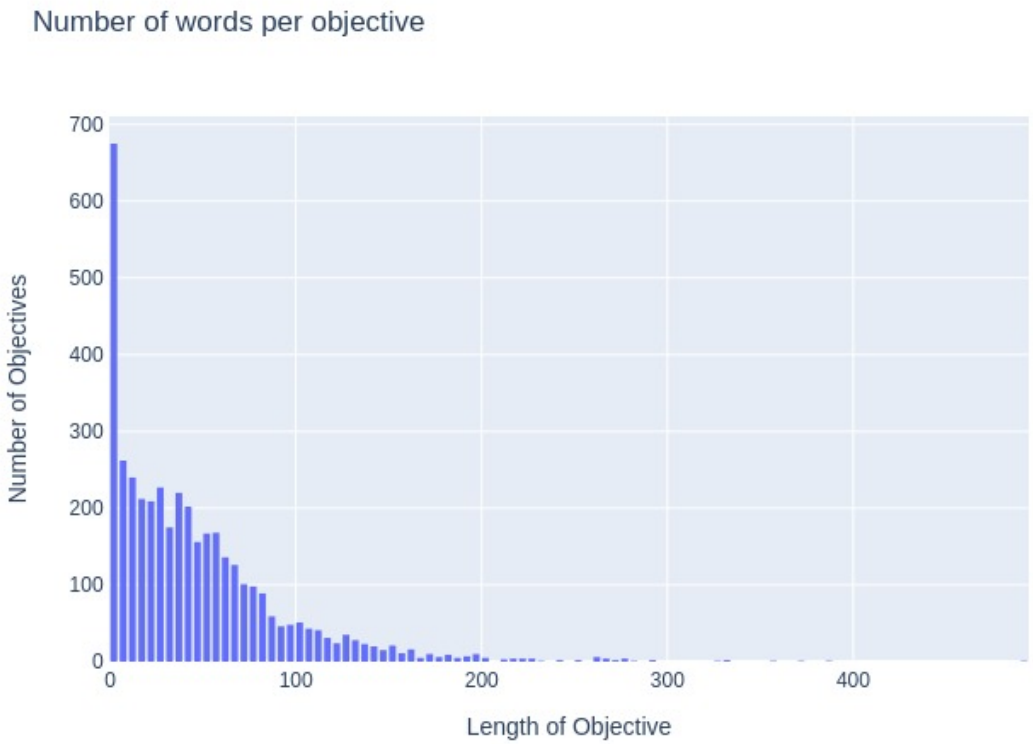


Figure 6.1: Statistical Analysis on the length of the objectives.

Analysis on language

Some of the language models used in this study are trained on English language. Hence, the dataset was filtered to keep only those objectives that were written in English language. Out of the 4078 objectives, there were 797 entries founds that were written in a foreign language. These objectives were dropped during the pre-processing steps and were not used in the Topic Modeling study.

Statistical analysis on number of objectives per employee

The dataset provided by the Merck Group contained multiple objectives for majority of the employees. A statistical analysis can be seen in figure 6.2 which shows that only 9 employees had 1 objective in the data While all the other employees had 2 or more objectives.

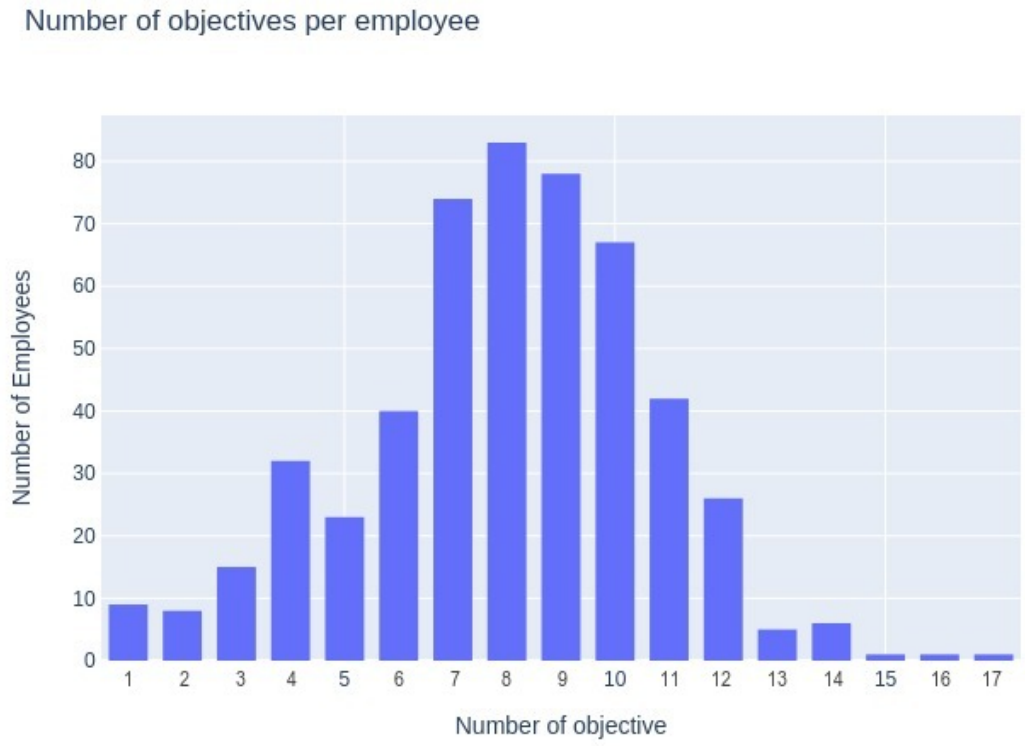


Figure 6.2: Statistical Analysis on number of objectives per employee.

Since the main focus on this study is to perform Topic Modeling on employee centric features, all the objectives belonging to an employee were concatenated to create a single entry.

6.1.2 Dataset Conclusion/Summary

The in-depth analysis of the dataset provided a lot of important information regarding it. Bases on that information, only the relevant columns were extract from the orignal dataset. Next, all repetitive objectives were removed and the remaining dataset was filter to remove the objectives with non-english text. The remaining objectives were truncated at length 219. As a final step, all objectives belonging to the same employee were merged into one. After performing all the above mentioned steps, the dataset had 511 entries. This dataset is very small to get an estimate over model parameters. Hence a secondary Job description dataset was used to fine-tune the model parameters. This dataset is discussed in more details in the next section.

6.2 Job Description dataset

Kaggle is a Data Science platform which contains many open source dataset for various application. It contains a benchmark Job description dataset with more than 200,000 entries. A sample data point from this dataset can be seen in table 6.3 . Since it a public dataset, no text has been redacted. However, the columns relevant to this study are mentioned in table 6.4. The main advantage of using dataset over the Employee Objective dataset is the number of samples as it allows to better generalize and draw conclusions regarding model parameters.

Table 6.3: A sample data point from the Job Description Dataset.

Column Name	Sample Data
Id	12612628
Title	Engineering Systems Analyst
Full Description	Engineering Systems Analyst Dorking Surrey Salary ****K Our client is located in Dorking, Surrey and are looking for Engineering Systems Analyst our client provides specialist software development Keywords Mathematical Modelling, Risk Analysis, System Modelling, Optimisation, MISER, PIONEEER Engineering Systems Analyst Dorking Surrey Salary ****K
Location	Dorking
Contract Time	permanent
Contract Type	full_time
Company	Gregory Martin International
Category	Engineering Jobs
Salary	20000 - 30000/annum 20-30K

Table 6.4: Job Description dataset with selected features.

Column Name	Column Description
Id	This column contains the anonymous job id
Title	This column contains the Job Title
Full Description	This column contains the Job description

6.2.1 Dataset Analysis

This Job description dataset is a benchmark dataset from Kaggle and is hence easier to use as it requires minimal pre-processing. The quality of text in this dataset is comparatively better as can be seen in analysis below:

Redundancy in Objectives

The dataset contains 244,769 job description with no redundancies at all.

Statistical Analysis on Job description length

A statistical analysis on the job description dataset is illustrated in figure 6.3 which shows the summary of number of words in each job description. The shortest length of a a description is 20 words. This means there is some description available for each job which is not the case for Employee Objective dataset. The longest job description contains 2118 words. Moreover, there are very few objectives that have length of greater than 700 words. Statistically speaking, 99% of the objectives have length equal to or less than 800 words. Similar to the Employee Objective dataset, all job descriptions with length greater than the maximum among the 99% of the job descriptions were truncated to 709 words.

Analysis on language

All the job descriptions are in English Language so no entries were dropped.

6.2.2 Dataset Conclusion/Summary

An analysis on the Job description dataset has proven it to be of good quality. It requires almost no iterations except truncating the description at maximum length of 99% of the descriptions which is only done for faster computation as language models require all text to be padded to match the maximum length. Since there is only one job description per id, there is no need to merge job descriptions.

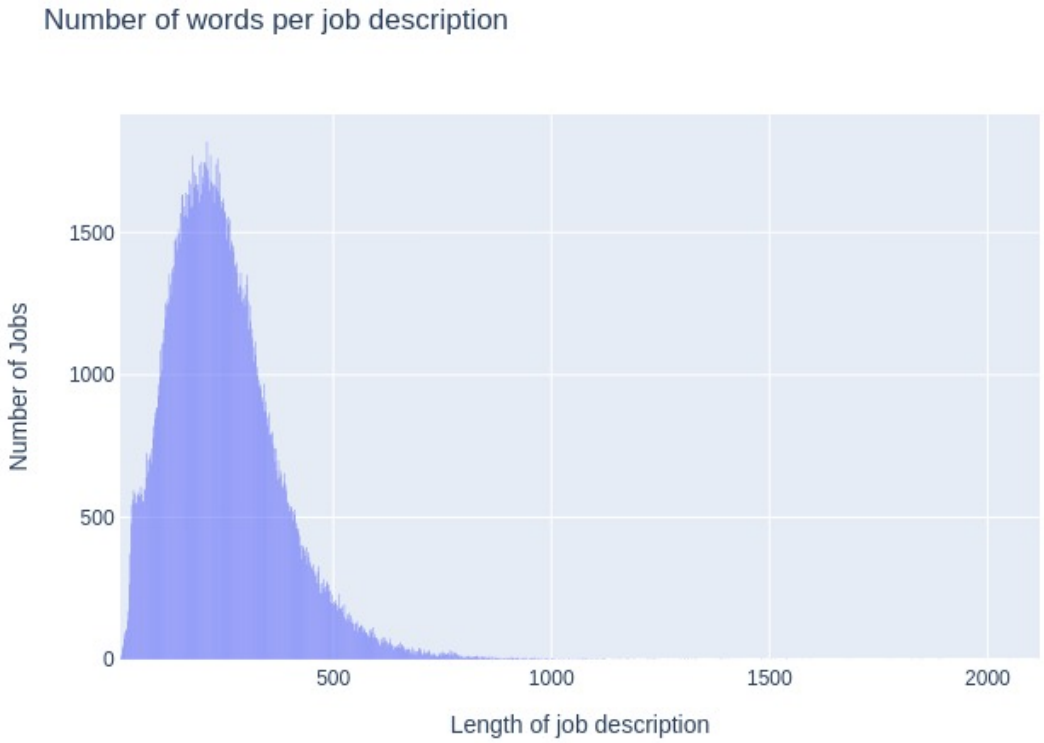


Figure 6.3: Statistical Analysis on the length of job descriptions.

7 Experiments

This study focuses on a Topic Modeling task on the Employee Objective dataset whose methodology is already discussed in chapter 5 in an abstract form. This chapter discusses the configurations and technical of the model parameters defined in the methodology. The general idea is to try out many different combinations and compare the results using human evaluation of word clouds and statistical measures. While this chapter focuses on the technical details of the experiments, the results are discussed in next chapter.

7.1 Pre-processing

Pre-processing is a very essential part of any text modeling tasks as it prepares the data in a form that is understood by the models. This section explains the configurations of the text Pre-processing pipeline defined in section 5.1.

7.1.1 Text Cleaning

Text cleaning is a noise removal process used to enhance the feature space. Following steps were used in this study to clean the texts:

1. Includes missing delimiters in the text ('.. good.This is okay' -> '.. good. This is okay').
2. Converts the text to lower case (AMaziNg -> amazing).
3. Remove letter repetition is it repeats more than two times (goood -> good).
4. Remove all special characters ('#hello' -> 'hello').
5. Eliminate phrase repetition ('This is very very good' -> 'This is very good').
6. Remove all numeric characters ('I want to eat 2 pizzas' -> I want to eat pizzas)

7.1.2 Tokenization

Natural Language Toolkit (NLTK) [63] is a text pre-processing library which provides a function to perform Tokenization. In this study, NLTK's tokenize function was used to convert sentences into list of words. This standard tokenization process works well for an LDA model however as already discussed in section 5.1.2, a model specific tokenizer should be used for Word Embedding Models from huggingface [55] library. For example, for a BERT [3]

model, huggingface provides a BertTokenizer to be used for pre-processing the text. Sentence Transformers such as Sentence Bert [4] do not require this tokenization step as they accept text as an input.

7.1.3 Encoding

For an LDA Model, the standard practice is to remove all stopwords from the tokenized words. In addition to that, all the non-noun words were also removed in this study. As an additional step, all the words were lemmatized using the NLTK library. The general idea is to remove the morphological affixes from words which is known as stemming. This can often times lead to incorrect words. In contrast to the stemming, lemmatization is better approach as it converts the word to its meaningful base form. Since an LDA model is not capable of processing the grammatical context of the words, it might treat 'run' and 'running' as two different words. To avoid this, words are lemmatized to avoid redundancies. After lemmatization, gensim [57] library is used to create a dictionary and encode the word lists. For the Word Embedding model, the same tokenizer from tokenziation step can be used to encode the text. The encoding step is not required for Sentence Transformers.

7.2 Feature Selection

There are three main source for feature extraction from text; LDA model, Word Embedding models and an Autoencoder [29]. As already depicted in figure 5.1 in the Methodology Chapter, five types of features can be constructed using different combinations of the main sources. A summary of these combination is presented in table Table 7.1. The first feature is the latent representation of a Word Embedding model features obtained using an Autoencoder. The second type uses the raw features from Embedding model. Third feature is the concatenation of the Word Embedding model features with LDA [1] model features. The fourth type learns the latent representation of the third features with using an Autoencoder. The last and fifth type is the document-probability vector which is used as the baseline.

Table 7.1: Possible combination amongst features to create a new feature space to be used for clustering.

Feature id	LDA	Word Embedding	Auto Encoder
1	x	✓	✓
2	x	✓	x
3	✓	✓	x
4	✓	✓	✓
5	✓	x	x

7.2.1 Latent Dirichlet Allocation Features

An implementation of LDA [1] model is provided by gensim [57] library which is used in this study. The results from this model is used as a baseline to compare results from all other techniques.

7.2.2 Word Embedding Features

Word Embedding features are extracted from three types for models as discussed below:

1. A general word embedding model returns an embedding of size [l x m] where 'l' refers to the number of tokens in the sentence and 'm' is the embedding size. A standard practice to obtain a sentence level representation by average over the features of all tokens. This approach is also used in this study for the transformer models from huggingface [55] such as XLNET [5] and XLM [59].

2. Some of the hugging face models such as BERT [3], and ELECTRA [60] were trained for classification tasks. Because of this, they have a special ['CLS'] token that provides a sentences level representation. This ['CLS'] token is appended at the beginning of the sentence by the tokenizer during the pre-processing stage.

3. Sentence Transformers models which take the original text as input and outputs the sentence level embedding of size 'm' where 'm' is the model specific embedding dimension. A Sentence Bert [4] base model and RoBERTa [61] model have embedding dimensions of 1024 while DistilBERT [62] has an output of dimension 768.

7.2.3 LDA Features + Word Embedding Features

The output from an LDA Model is a [N x k] matrix where 'N' is the number of documents and 'k' is a hyperparameter which refers to the number of clusters. Each row of this matrix gives the probability distribution of each document over the clusters. The output from the Embedding model is a [N x m] matrix where 'N' is again the number of documents and 'm' is the size of embedding features. In this techniques the features the both sources are concatenated to create a matrix of size [N x k+m] which includes the global and local information of the documents.

7.2.4 Auto Encoder

Since the feature space from the embedding vectors is very high dimensional, it is possible that it contains many unnecessary features. An Autoencoder [29] is used to learn a latent representation of the high dimensional features spaces such as the one obtained from Word Embedding models. Essentially in addition to Embedding features, and LDA features in tandem Embedding features, two new features spaces are created by passing both of them through an Autoencoder. The specification of the Autoencoder are discussed in section 5.2.4. Additionally the weights of the autoencoder were initialized using xavier [64] initialization

and trained using a SmoothL1 loss function which was introduced in RetinaMask [65]. The purpose of this experimentation is to evaluate if using a latent representation of the original feature vector could lead to better results.

7.3 Topic Modeling

For Topic Modeling to work, it is important to find the correct value of 'k' for clustering as well as the most effective technique for topic word retrieval. However, these two go hand-in-hand because the most reliable way of evaluating the results is through human evaluation of the topic words in the form of word clouds. This is because the word cloud provides a way to visualize the clustering and topic words at the same time. To fine-tune these model parameters, a series of experiment were conducted whose details are discussed in the sub-sections below:

7.3.1 K-means Clustering

K-means clustering is used to cluster the feature vectors into similar groups. For K-means clustering to work properly, it is essential that the value of 'k' is correct. Since this value is not known in advance, the experiments were conducted on values of k = 2, 3, 4, 5, 6, 7, and 8. The values of k are capped at 8 because on values larger than 8, the some clusters didn't contain any unique topic words. In addition to the five types of feature spaces, experiments were also conducted for various values of k. It is very essential to experiment with values of k and find the correct one. This is because of the dependency of Topic words techniques on the clustering algorithm. If the objectives are not clustered properly, then the topic words belonging to a cluster would never be coherent as they would belong to objectives that are very different from each other. For a specific value of 'k', If distinct themes are visible in the word clouds then it is an indicator that the value of 'k' is good. However, it is also important to increase the values of 'k' to evaluate if new themes appear in the new clusters.

7.3.2 Topic Words

It is important that Topic word retrieval is done correctly because it helps in evaluating the whole Topic Modeling pipeline. Topic words retrieval involves two parts; first is the technique using which topic words are found and second is the N-gram which defines the number of words in each topic.

Techniques

There are three types of techniques that are used to get topic words for the clusters. The configurations for each of them are discussed below:

1. Transformer Attention: As discussed in section 5.3.2, self Attention mechanism of a Transformer model provides a matrix that depicts how much attention the model should pay

to the all the previous words when processing a given word. The general idea is to iterate through all the transformer layers and sum up the attention value assigned to each unique token and use the tokens with highest aggregated attention value as the topic words. This approach is discussed in more details in the steps below:

1. Initialize a dictionary to save tokens and their respective attention value.
2. For each text in cluster, use the same tokenized representation as the one used to obtain Embedding features in section 7.1.2.
3. Filter out the tokens for stopwords and words containing '###'.
4. For each remaining token, sum up all the attention paid to that token by all the other tokens in the sentence.
5. If the word doesn't already exist in the dictionary then include this token and it's collective attention value in the dictionary. Otherwise, update the attention value of the existing entry for this token by adding this new attention value to the existing one.

2. TF-IDF TF-IDF is a keyword ranking algorithm that can be used to find topic words within clusters. In this study, the TF-IDF implementation by gensim [57] library is used as follows:

1. Using the cleaned texts from section 7.1.1, for each cluster, concatenate all the texts belonging to it into one document. This would give k documents.
2. The k documents are passed through a gensim [57] implementation of TF-IDF model which assigns a weight to each word in the cluster depicting it's importance.
3. For each document, Top N words with highest weights are picked as the topics for that cluster.

3. Frequency-based A frequency-based approach is the simplest approach for finding Topic Words and it is used as a baseline for comparing the results from TF-IDF and Transformer Attention [15] approach. It operates on the assumption that words with more frequency are important and can be used as topic words. Following steps are used to find the topic words using the frequency approach:

1. For each cluster, group all the tokens from the texts belonging to it. For consistency, use the tokens from the tokenization step discussed in section 7.1.2.
2. All the stop words and non-noun words are filtered.
3. A dictionary is created which has the 'words' as keys and their occurrence frequency as the value.
4. Top N words with maximum frequency are selected as the topic words for that cluster.

N-grams

Instead of using only single words known as 1-grams to define the topic, it is also possible to use phrase of two or three words known as 2-gram and 3-gram respectively. Experiments were conducted with configurations such as various values of N-grams. Given a token list that maintains the order of words from the original sentence, topic words or phrases for any N-gram can be obtained using the following steps:

1. initialize a dictionary to save tokens and their counts.
2. For each cluster, iterate over the ordered list of tokens while filtering out stopwords and non-noun words depending on the configurations.
3. Create a phrase by concatenating N consecutive words in the forward order. For example, for a 2-gram approach, the sentence 'Everyone should eat one apple a day to stay healthy' would have features ['Everyone should', 'should eat', 'eat one', 'one apple', 'apple a', 'a day', 'day to', 'to stay', 'stay healthy']. Whereas if the tokens are filtered for stopwords, the features would be ['Everyone eat', 'eat one', 'one apple', 'apple day', 'day stay', 'stay healthy']
4. For each phrase, if the create phrases already exist in the token dictionary, then update the count by one. Otherwise creating a new entry with count one.

7.4 Post Processing

The clusters formed by Topic Modeling techniques often time contains some common words. This creates overlaps of topic words within clusters which doesn't align well with the goal of this study. The idea is to get more coherent clusters where each word or phrase could belong to only one cluster. Two policies were introduced in the section 5.4 those results are then compared with the results before Post-Processing. For the evaluation of the topic words two approaches are used; First is the human evaluation of the word clouds and second is the coherence score calculated before and after the post processing step on topic words belonging to each clusters.

7.4.1 Hard Policy

The Hard Policy deletes the overlapping topic words from all clusters using the following steps:

1. For all clusters, get a unique list of all overlapping topic words or phrases.
2. Delete those words for the topic words list of all clusters.

7.4.2 Soft Policy

The hard policy has a downside as it deletes topic words entirely. This causes loss of meaningful information. The Soft Policy on the other hand keeps the redundant words in the cluster that has the maximum frequency of the topic word using the following steps:

1. For each cluster, initialize a list of topic words to be delete
2. For each token in a cluster, iterate through the topic words of all other cluster to see if a redundancy exists.
3. If a redundancy is found, then compare the frequency of the token in both clusters. If the frequency in current cluster is lower than the topic word , it is added to the to delete list.
4. Once step 1 has occurred for all clusters, the words in their respective to delete lists are deleted from the topic words of all clusters.

8 Results and Discussion

The previous chapters have discussed the Methodology followed in this study, details of the dataset as well the technical details of the experiments. This cshapter builds up on the information from previous chapters and discusses results from experiments mentioned in chapter 7. However, before comparing results from various experiments, it is important to have an appropriate measure along which the model performance is analyzed. Theoretically, Topic Modeling has two parts, Clustering and Topic Words retrieval for which one analytical measure each is used. Furthermore, one human evaluation metric is used in this study to record the performance for both tasks. First analytical measure is Coherence score [66] which is used to analyze how similar the topics are within a cluster. This is used to evaluate the consistency of topics selected for all clusters. Second analytical measure is a Silhouette score which is a measure to evaluate the performance of K-means algorithm. Since the dataset used in this study is an unsupervised one, the analytical measures are not reliable as there is a lot of room for error. For this reason, a third evaluation measure called Word Cloud is used. A word cloud is discussed in the details in section 4.1.2 but essentially it displays the topic words for each cluster in an image which allows to evaluate the results of both clustering and topic words. The drawback of this measure is that it requires human input which is time-consuming. Details of these measures are discussed below:

Coherence Score: A Coherence score [66] is used to measure the degree of similarity between topics in a cluster. The general idea is that if the topics or phrase support each other, it results in a higher coherence score for that cluster. Topic Coherence calculates this by evaluation the semantic similarity between top words in the cluster. This study used implementation of Coherence Model from gensim [57] library to calculate Coherence Score. The measure used in this study for calculating coherence is 'C_v' which works with a sliding window, one-set segmentation of the top words and uses Cosine Similarity and point-wise mutual information (NPMI) as an indirect confirmation measure. This outputs a value between 0 and 1. A value closer to 1 means that the model is doing well and topics within the cluster are well coherent. Getting a value between 0.8 and 1 is unrealistic. A value between 0.5 to 0.7 is a good indicator where values around 0.4 and lower is considered a bad score.

Silhouette Score: K-means clustering works with unlabeled data which entails that no ground truth is available to assert the correctness of the method. There are however some intrinsic methods which can be used to evaluate the clustering quality. Silhouette Score being one of them examines the compactness of the data point features within a cluster and how well the clusters are separated from each other. This study uses the implementation provided

by scikit-learn [67] library which calculates Silhouette Coefficient for each sample. It outputs a value between +1 and -1. A score closer to +1 indicates that the sample is far away from the points in the neighboring clusters while a value closer to -1 indicates that the sample might have been assigned to the wrong cluster. A value closed to 0 however means that the sample is either very close to or on the decision boundary between other clusters. The final score is the mean Silhouette Coefficient of all samples.

Word Cloud: Word Cloud is a visualization which shows the results of clustering and topic words combined. It generates 'k' images where k is the number clusters and each images contains topic words belonging to one cluster. A python word cloud library is used to generate these images which requires the topic words along with their frequencies as input. It allocates the font size to each word depending on it's importance. In this study, the minimum font size allowed is 7 and all words that are allocated this size are not include in the word clouds.

8.1 Overview

The main contribution of this study is experimentation with various types of feature spaces for a Topic Modeling task. As a first step, results were analyzed for various Embedding models. To that that experiments were conducted with seven different Embedding models whose word clouds are discussed in details in the later section of this chapter. Coherence Scores [66] and Silhouette Scores for all the embedding models are shown in the table 8.1. According to these statistical measures, XLM [59] model has the best Coherence Score and XLNET [5] has the best Silhouette Score. Both of these are contradictory to the Human evaluation of word clouds according to which the Sentence BERT [4] provides the best results.

Table 8.1: Coherence scores and Silhouette Scores when using feature space from Embedding Models for clustering before post-processing step with 10 clusters, frequency-based topic word retrieval approach on Job Description dataset

Model	Coherence Score	Silhouette Score
Sentence BERT	0.5333	0.0638
BERT	0.5857	0.1321
XLNET	0.4938	0.1324
Sentence RoBERTa	0.5499	0.0427
ELECTRA	0.5743	0.1169
Sentence DistilBERT	0.6040	0.0435
XLM	0.6118	0.0744

Next, experiments were conducted for five feature spaces with Sentence BERT [4] as the default Embedding mode. First is the probability vectors from the LDA [1] model which is used as the baseline approach. The second and third include feature vectors from Sentence

Bert Embedding model, LDA + Sentence Bert embedding model, While the fourth and fifth include latent spaces of features from second and third respectively. All of these results are also discussed in the later section of this chapter while the Coherence Scores and Silhouette Scores are mentioned in table 8.2.

Table 8.2: Coherence scores and Silhouette Scores when each feature space is used for clustering. Score are captured before post-processing step with 10 clusters, frequency-based topic word retrieval approach on Job Description dataset.

Model	Coherence Score	Silhouette Score
LDA	0.4629	N/A
Embedding Model	0.5333	0.0638
Embedding Model+ LDA	0.6001	0.0745
Embedding Model+ Autoencoder	0.6280	0.1392
Embedding Model+ LDA + Autoencoder	0.6083	0.2456

Using the Sentence BERT model as the default feature space along with best model parameters as defined in table 8.3, a Topic Modelling framework was designed for to Merck Group. A target group from Merck evaluated the framework and provided a survey. The summary of results with respect to each evaluation aspect is shown in figure 8.1.

Table 8.3: Summary of best configurations for model obtained for Topic Modelling on Job Description Dataset.

Experiments	best configuration
Embedding Model	Sentence BERT Model
Feature Space	Embedding Model
Topic Word Technique	Frequency based, TF-IDF
Post Processing policy	Soft Policy
Number of clusters (k)	12
N-gram	1-gram

8.1.1 Research Questions

Could using embedding vectors lead to better results than Latent Dirichlet Allocation model?.

Yes, using a Word Embedding model for Topic Model can lead to better results.

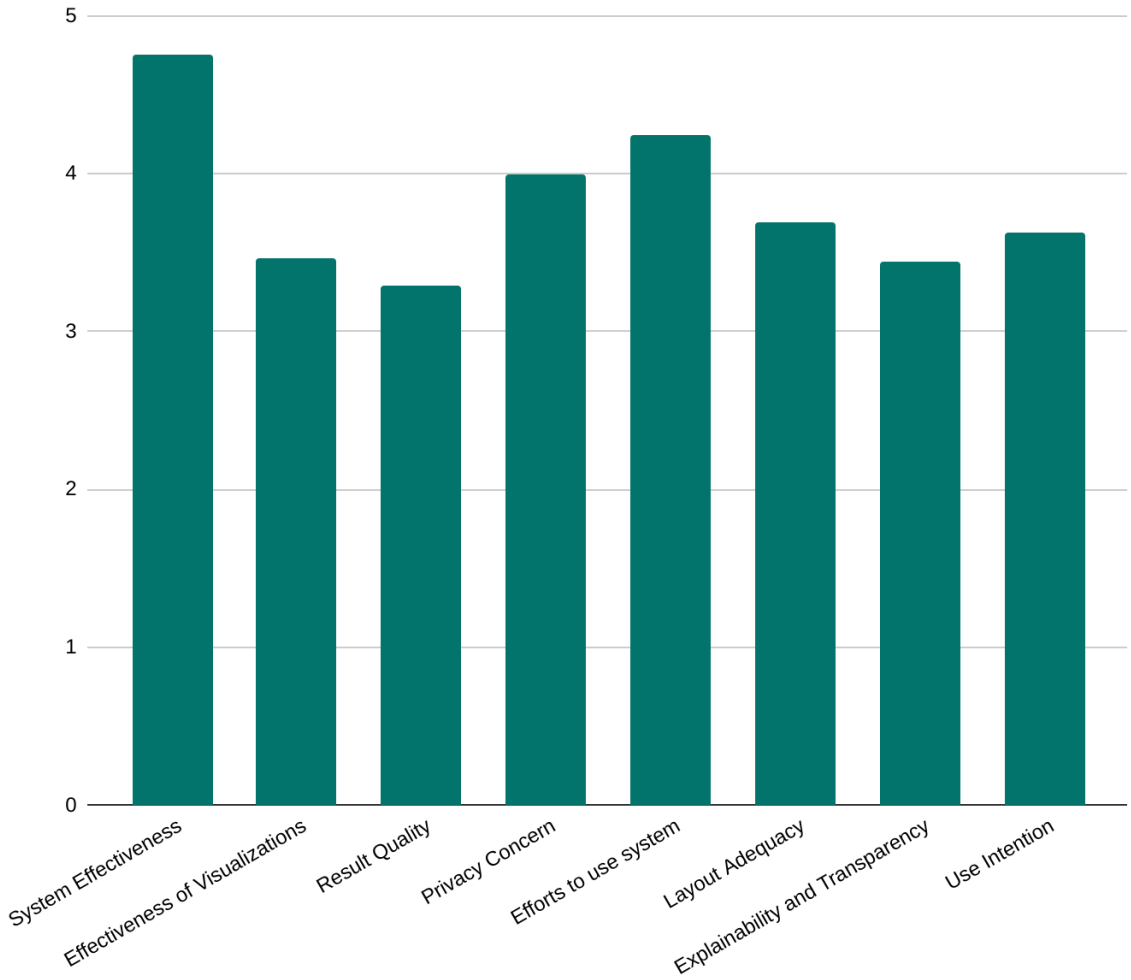


Figure 8.1: A summary of feedback from Merck Group on the Topic Modeling Framework. On the x-axis, each histogram represents an evaluation aspects which is demonstrated by their labels. The participant expressed their agreement/disagreement for each aspect on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree] which is depicted on the y-axis.

If the word embedding models are able to provide better results, then which type of embedding model is better suited?.

More specifically, Sentence Transformers Models such as Sentence BERT [4], Sentence RoBERTa [61] and Sentence DistilBERT [62] provide the best results.

Could using a traditional algorithm such as LDA in tandem with the Embedding models provide better results?.

No, using Word Embedding in tandem with an LDA model provide almost the same result as using a Word Embedding model alone.

8.2 Results on Job Description Dataset

As discussed in chapter 6, Employee Objective dataset only has 511 entries left after removing duplicates and foreign language text. Furthermore, this dataset contains objectives of employee mostly from the HR department at Merck Group. Because of this, the data does not contain that many clusters and is hence not suitable enough to draw general conclusion about model's behaviour. A second Job Description dataset is used to conduct the initial experiments to find best configurations for model parameters. The results experiments are broken down into five groups. First section discusses the results from the five types of features spaces and finding the one with best performance. Next is using the selected feature space to discuss the performance of three types of Topic Word retrieval techniques. The third section shows results before after the Post Processing step and discusses the results from hard policy and the soft policy. Fourth section performs Topic Modeling tasks by clustering using various values of 'k' and The goal is find the value of k that works best. The last section shows results for 1-gram, 2-gram and 3-gram. The results shown in this section were generated using randomly selected 50,000 samples from the Job Description dataset. Once all of the configurations have been selected, they are used in next section which discusses the results on Employee Objective Dataset.

8.2.1 Experiments with Feature Spaces

This sub-section discusses the results generated by the five types of features depicted in figure 5.1. Since the best configurations for other parameters in the model are not known yet, the most basic configurations such as the n-gram = 1 and frequency-based techniques for topic words is used. However, before moving forward with the evaluation of feature space, it is essential to find out which type of embedding model works best. The reason for doing this it keep the experiments easier to follow and discuss.

Topic Modeling using embedding model features

Word Embedding models provided multi-dimensional context driven representation for texts. There are many state-of-the-art Word Embedding models available. Therefore, before experimenting with various type of feature spaces, it is essential to find an Embedding model that works best for the Topic Modeling task. Results for various types of Embedding models are discussed below.

1. Sentence BERT The word cloud generated using Sentence Bert [4] embedding model is shown in figure 8.2. Even though there are some redundant topic words between clusters such as 'experience' in clusters [1, 2, 4, 5, 6, 9] and 'sale' in clusters [4, 5], it is still possible to recognize the theme of the jobs grouped together. The reason for some redundant topics in clusters is that the words such 'experience', 'sale' and 'team' are common in all job descriptions and hence it makes sense why they would have high frequencies. The coherence score for these topic words is 0.533 which is good enough. The Silhouette Score however is 0.0638 which close to 0. It is an indicator that samples in all clusters are close to each other within their feature spaces. This could be because there many common words and stop words in all job descriptions.

2. BERT The results from Bert [3] (base) model are shown in figure 8.3. Even when neglecting the common topic words, it is clearly seen that themes formed by clusters do not make any sense. The topics words picked by all clusters are almost same for example the words 'experience', 'sales', 'work' 'business', 'team'. One possible reason for this could be that the clustering is not done accurately which is leading to these ambiguous clusters. Coherence score of 0.5857 and Silhouette Score of 0.1312 is higher than Sentence Bert model which is unusual because the topic modeling results according to human evaluation of word cloud are worse than Sentence Bert.

3. XLNET Word clouds obtained using the feature space from Xlnet [5] model are displayed in figure 8.4. Similar to Bert model, the clusters obtained using Xlnet features are very ambiguous and do not contain any theme. Most clusters contain same words such as 'manager', 'skills', 'work', 'job', 'business' which are main key words for any job description. Similar the Bert [3] model, the Coherence score of 0.4938 and Silhouette score of 0.1324 contradict the human evaluation of Word Clouds. These scores indicate that the model is doing better than others but this is not the case.

4. Sentence RoBERTa The results for Sentence Roberta [61] can be seen in figure 8.5. Similar to Sentence Bert, the clusters show coherent topics which are suppressed by the noise in terms of common topic words. Coherence score of 0.5499 is in the acceptable range however the Silhouette Score of 0.0427 contradicts human evaluation. Such low value of Silhouette score could be because of the common topic words in clusters as embedding features would be close to each other in many dimensions.

5. ELECTRA Word clouds obtained using the feature space from Electra [60] model are displayed in figure 8.6. Similar to Bert model and Xlnet model, the clusters obtained using Electra features are very noise and do not depict any coherent theme. Both the Coherence score (0.5743) and Silhouette score (0.1169) contradict human evaluation.

6. Sentence DistilBERT Distilbert is a light-weight version of BERT [3] Embedding model. The results for Sentence Distilbert [62] can be seen in figure 8.7. The word clouds generated

Figure 8.2: Word Clouds when Sentence BERT Embedding model is used as feature space for clustering before post-processing step with 10 clusters, frequency-based topic word retrieval approach on Job Description dataset.

(a) Cluster 1.

(b) Cluster 2.

(c) Cluster 3.

(d) Cluster 4.

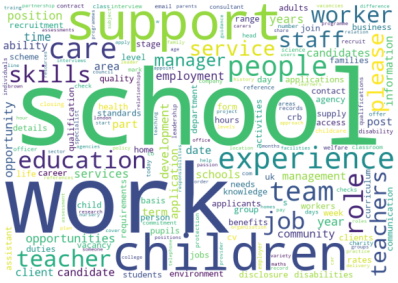
(e) Cluster 5.

(f) Cluster 6.

(g) Cluster 7.

(h) Cluster 8.

(i) Cluster 9.



(j) Cluster 10.

Figure 8.3: Word Clouds when BERT Embedding model is used as feature space for clustering before post-processing step with 10 clusters, frequency-based topic word retrieval approach on Job Description dataset.

(a) Cluster 1.

(b) Cluster 2.

(c) Cluster 3.

(d) Cluster 4.

(e) Cluster 5.

(f) Cluster 6.

(g) Cluster 7.

(h) Cluster 8.

(i) Cluster 9.

(j) Cluster 10.

Figure 8.4: Word Clouds when XLNET Embedding model is used as feature space for clustering before post-processing step with 10 clusters, frequency-based topic word retrieval approach on Job Description dataset.

(a) Cluster 1.

(b) Cluster 2.

(c) Cluster 3.

(d) Cluster 4.

(e) Cluster 5.

(f) Cluster 6.

(g) Cluster 7.

(h) Cluster 8.

(i) Cluster 9.

(j) Cluster 10.

Figure 8.5: Word Clouds when Sentence RoBERTa Embedding model is used as feature space for clustering before post-processing step with 10 clusters, frequency-based topic word retrieval approach on Job Description dataset.

(a) Cluster 1.

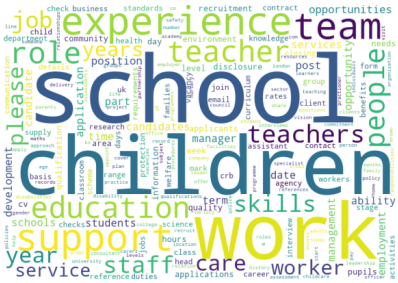
(b) Cluster 2.

(c) Cluster 3.

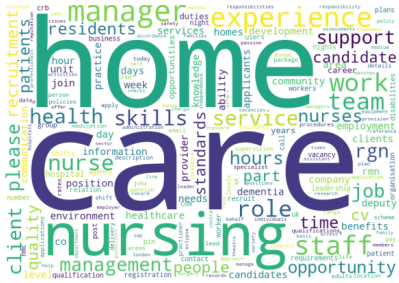
(d) Cluster 4.

(e) Cluster 5.

(f) Cluster 6.

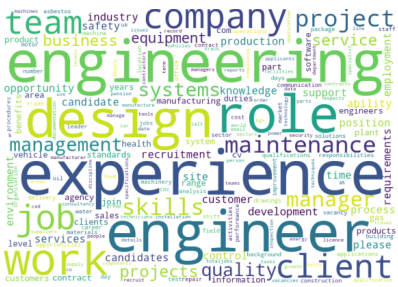


(g) Cluster 7.



(h) Cluster 8.

(i) Cluster 9.



(j) Cluster 10.

using features vectors obtained from Sentence Distilbert model are similar to other Sentence embedding models. The clusters show some themes which is suppressed by the common topic words. The coherence score is a bit higher than other Sentence Embedding models with a value of 0.6040. The Silhouette Score is 0.0435 which very low and also contradicts the human evaluation. As clearly seen in the word clouds, the jobs are clearly separated from which each and hence a low Silhouette Score is not justified.

7. XLM The word cloud generated using Xlm [59] embedding model can be seen in figure 8.8. The result are a little better than other word embedding models such as Xlnet and Bert. Even though most clusters are noisy, clusters [3, 8, 10] depict a weakly formulated theme. Coherence score of 0.6118 is highest for this model which does not align with the noisy topic words seen in Word Clouds. Similar to all other models, Silhouette Score is close to zero with a value of 0.0744.

Results from various features spaces

The previous section discussed the results generated from different Embedding Model. Among all the word embedding models, Sentence Bert has the best performance while Sentence Roberta has the second best. For the experiments discussed in the upcoming sections, Sentence Bert [4] would be used as the default embedding model. As discussed in the chapter 5 Methodology, there are five types of features spaces whose results are to be evaluated in this section.

1. LDA Model (Baseline): An LDA model is a probabilistic approach used for Topic Modeling tasks. The results from the LDA model on Job Description dataset are shown in figure 8.9. Amongst the cluster, some of the Word clouds portray a consistent theme such as clusters [2, 3, 6, 8] but there are clusters which are very noisy such as clusters [1, 4, 5]. On a positive note, there are no redundant topics words in between clusters. The Coherence score for the LDA model is good but less than all the feature spaces. Since an LDA model uses a probabilistic approach for separating document, Silhouette Score is not applicable.

2. Embedding Model (Sentence BERT): In the previous section, feature space from a Sentence Bert [4] model provided best results. Therefore, it is used as the default Embedding model in this study. The results from the Sentence Bert model are shown in figure 8.2. As already discussed before in section 8.2.1, all the clusters form a coherent theme with the exception of the common topic words. The Coherence Score (0.5333) and Silhouette Score (0.0638) are lowest when compared to other feature spaces.

3. Embedding Model (Sentence BERT) + LDA: One research question in this study is evaluate if combining results of an Embedding Model with LDA can lead to better performance. The results from the Sentence Bert model with LDA are shown in figure 8.10. The theme formed by word clouds using this feature space is less consistent compared to Sentence

Bert but still good enough. Clusters [3, 6, 8, 9, 10] have clear visibility of themes. There redundancy in topic words also exists in this case. The Coherence score is above average with a value of 0.6001 while the Silhouette Score is 0.0745.

4. Embedding Model (Sentence BERT) + Autoencoder: The results obtained using the latent feature representation of Sentence Bert model are shown in figure 8.11. Word clouds generated using this feature vector is quite similar to the ones generated by using the original feature space. The Coherence score is highest for this feature space with a value of 0.6280. Silhouette Score is 0.1392 which is better than the other models but still not good on a global scale. The reason for this increase in Silhouette score is the use of Autoencoder. It provides a latent representation of the original feature space between 32 to 64 dimensions. This lower dimension is easily separable by k-means algorithm compared to the much higher dimension in the original space which results in a better Silhouette Score.

5. Embedding Model (Sentence BERT) + LDA + Autoencoder: This section discusses the results obtained using the latent representation of feature space constructed by concatenating Sentence BERT feature vector with LDA model. The word clouds generated from the Sentence Bert model with LDA and Autoencoder are shown in figure 8.12. The results are similar to all the other other models with the exception of LDA model. The coherence score is 0.6083 which is a good indicator while the Silhouette Score is 0.245 which is the highest among all models. The reason for this increase in score would be because of the usage of probability vector from LDA model which makes it easier for k-means to separate clusters.

8.2.2 Topic Word Retrieval

The previous section shows that the word clouds generated from the four type of feature spaces are almost identical and different exists only in the Analytical Scores. For the sake of simplicity, feature vectors from Sentence BERT model are used as the default feature space. Moreover, the results obtained from Sentence Bert Embedding model gives better results compared to an LDA model. In this section, the results obtained using the three types Topic Retrieval techniques are discussed. The coherence scores for these techniques are displayed in table 8.4.

Table 8.4: Coherence scores for Topic Retrieval techniques with feature vectors from Sentence BERT Embedding model for clustering before post-processing step with 10 clusters on Job Description dataset.

Techniques	Coherence Score
Frequency-based	0.5333
self Attention	0.4685
TF-IDF	0.3391

1. Transformer Attention Transformer [15] models are trained using an extra self Attention layer which tells the model how much weight should be assigned to all previous words when predicting a new word. The word cloud generated using self Attention from Bert [3] model are shown in figure 8.13. It can be seen that the topic words for each cluster are inconsistent despite the fact that there are no common topic words between clusters. The reason for these inconsistent theme is the fact that the clustering is not accurate. As already discussed in section 8.2.1 that the sentence level feature representation obtained using the huggingface [55] transformer models is not meaningful and hence leads to imprecise clustering. A Topic retrieval method can only be effective if the documents are clustered correctly. This is not the case here. The accurate clustering is only provided by the Sentence Embedding models such as Sentence Bert or Sentence Roberta but these model do not contain a self Attention layer. The coherence score is in the average which doesn't make sense as the clusters are not coherent at all.

2. TF-IDF TF-IDF is a classical NLP technique for finding important keywords in a document. The word clouds obtained when using TF-IDF as a topic words retrieval techniques are shown in figure 8.14. All the clusters form a coherent theme with the exception of the common topic words. This technique has the lowest Coherence score. Despite the fact that the coherence score is lower than the frequency based approach, the performance with respect to human evaluation is almost the same.

3. Frequency-based This technique assigns score to words based on their frequency in the document. It also filters out the words for stopwords. The results from Frequency-based approach as a topic words retrieval techniques are shown in figure 8.2 and discussed in section 8.2.1. This approach has the highest Coherence score of 0.5333.

8.2.3 Post Processing

In this study, so far all the Word Clouds obtained are noisy. There is a high degree of redundancy between the topic words of clusters. Even though there is a good explanation for these redundancy such as the fact that the redundant words are common keywords for any job description and would hence be qualified as a top words in most cluster. But, they act as noise and mask out the main topic words which could help form a coherent theme in clusters. This post processing step is used to filter out this noise by removing these common words. There are two techniques used for this noise removal. One deletes the common words entirely and the other one allows keep redundant word in the cluster which has the maximum frequency of that words. The results of experiments conducted using both these approaches are discussed below and their coherence score are shown in table 8.5. Even though the hard policy has a higher score of 0.7207 compared to the soft policy with 0.6567, the word clouds obtained through the soft policy are better suited for this study.

Table 8.5: Coherence scores before and after the post processing step with Sentence BERT Embedding models with 12 clusters, frequency-based topic word retrieval approach on Job Description dataset.

Policy	Coherence Score (Before)	Coherence Score (After)
Soft	0.5040	0.6567
Hard	0.5985	0.7207

Hard Policy

The Hard policy deletes all redundant words from all clusters. The results after using a hard policy are shown in figure 8.15. It can be seen in the word clouds that after removing the redundant words, the theme formed by clusters doesn't make any sense. The topic words in the images provide no information regarding the theme at all. Furthermore, as seen in table 8.5, the coherence score has also increased from 0.5985 to 0.7207 which contradictory to the human evaluation of the word clouds. Based on the results, it is clear that the hard policy is not a good approach to be used as the post processing and hence would not be used further in this study.

Soft Policy

A Soft policy keeps the topic words in the most relevant clusters. The results after using a soft policy is shown in figure 8.16. It can be seen that after removing the redundant words, the theme formed by clusters is much more visible. Furthermore, as seen in table 8.5, the coherence score has also increased from 0.5349 to 0.6557 which aligns with the human evaluation. This soft policy has helped bring out the theme within the clusters and would hence be used as the default configuration for the remaining part of this study.

It is possible that some of the embedding models that didn't work well in the previous experiments because of this noise. To double check the accuracy of the models from huggingface [55] library, this post processing technique with soft policy was also applied on the results from the Bert [3] (base) model whose word clouds are shown in figure 8.17. It can be seen that even after the post processing step, the clusters do not form a coherent theme with topic words. This confirm the theory that since the job description are not clustered accurately for Bert base and other models from huggingface library, the topic words are not coherent despite the fact that coherence score has increased from 0.5349 to 0.6567. This increase is coherence score could be an indicator that this measure is not reliable.

8.2.4 Number of clusters

The results of Topic Modeling depend heavily on the defined number of clusters 'k'. Since the dataset used in this study is unsupervised, several experiments were conducted with different values of 'k' to find the number of clusters that fit this dataset the best. The word

clouds for different values of 'k' are discussed in the section below and their Silhouette scores are shown in table 8.6. In the table, it can be seen that all the scores are close to zero, which means that all the samples in each cluster are very close to the decision boundary. This is true because the clustering is done using feature vectors from embedding model which contains many common words. Since the feature vectors are trained to capture the capture context across multiple dimensions, it is possible that values in many feature spaces are number close to each other. Furthermore, as already discussed before, there are many redundant topic words as well as stop words which exists in all clusters. Because of these reasons, it can be concluded that Silhouette Score is not a reliable measure for a task such as Topic Modeling using Word Embedding features.

Table 8.6: Silhouette scores for different values of k with feature spaces from Sentence BERT Embedding model used for clustering, after post-processing with soft policy, frequency-based topic word retrieval approach on Job Description dataset.

k	Silhouette Score
5	0.0751
8	0.0654
10	0.0638
12	0.0605
14	0.0583

k=5

When the number of clusters is set to 5, it can be seen in figure 8.18, that only 3 clusters [1, 2, 3] depict a weakly consistent theme. The remaining 2 clusters [4, 5] have divided all the remaining jobs between them which is depicted by their keywords which belong to various job description. The Silhouette Score is 0.075 which is low according to global standards of clustering.

k=8

When the number of clusters is increased to 8, it is depicted in figure 8.19 that clusters with new themes have emerged. Clusters [1, 6, 7, 8] portray new clusters of jobs with coherent topic words. Cluster 5 however still has an ambiguous theme which could be an indicator that the dataset has more clusters. Silhouette Score has decreased to 0.065 while the performance of Topic Modeling is better.

k=10

Word clouds for 10 job clusters are depicted in figure 8.16. Cluster [1,2,3,7,9,10] are same from previous experiment but cluster 4, 8 show new themes which were missing for k =

8. Silhouette score has further decreased but as seen in the Word Clouds, the quality of clustering has gotten better.

k=12

Word clouds for 12 clusters are shown in figure 8.20. As seen in the figure, the increase in number of clusters brought out more features of the dataset. Three new themes in clusters [5, 9, 10] have emerged which were not there before. Silhouette Score has decreased even further.

k=14

The results for 14 clusters are displayed in figure 8.21. Most of the themes are same as before except Cluster 13 which represents a new job theme. Silhouette Score for 14 clusters has further decreased to 0.058 which contradicts the word cloud images as the clusters are better separated.

8.2.5 N-gram

N-gram is a parameter in Topic retrieval techniques. Using the best configurations from previous sets of experiments, word clouds are generated for N-grams = 1, 2, and 3 and discussed in this section. Throughout this study N-gram = 1 was used during experiments. For reference purposes, the results can be seen in figure 8.20. Since best results were seen for k = 12, this value is used to test values of N-gram.

N-gram = 2

Word clouds were generated for 12 clusters and 2-gram whose results are displayed in figure 8.22. In most case, the 2-gram approach is able to create phrases which provided meaningful information such 'design-engineer', 'car-sales', 'sql-server' ,'head-chef' or 'nursing-home'. However, there are some instances when the meaning of 2-gram phrases is unclear and can cause confusion. Some examples for such phrases are 'agency-relation', 'placement-uk' or 'employer-brown'. But In general, the clusters are still able to depict a consistent theme but not as good as 1-gram.

N-gram = 3

The Topic phrases generated with 3-gram and 12 clusters is shown is figure 8.23. It can be seen that most clusters are not able to retain a clear theme and contain a lot of noises. There are few very rare cases when the phrases contain important information such 'html-css-java' but on a bird's eye view, most phrases cause more confusion. By observing the clusters it can concluded that 3-gram is not an effective approach for this topic modeling task.

8.3 Results on Employee Objective Dataset

This section focuses on Topic Modeling using the Employee Objective Dataset discussed in section 6.1. This section is divided into two parts, first part shows results on Employee Objective dataset and the second part discusses the survey filled by Merck Group.

8.3.1 Topic Modeling

The best model configuration obtained through experimentation are used in the NLP engine that serves as the back-end of the Employee Objective Framework. The framework has the number of clusters (k) as a configurable parameter and allows values between 3 to 8. One sample result group with 7 clusters is discussed below:

Word Cloud

Word Clouds generated with 7 clusters on Employee Objective dataset from Merck Group are shown in figure 8.24. Each word cloud in the figure depicts the main topic discussed by employees clustered in that group.

2D clusters

Using Sentence BERT model along with selected configurations, employees from Merck group were clustered on the basis of their objectives into 8 clusters. The 2D plot is shown in figure 8.25. This plot shows each employee as a data point in this scatter plot and depicts the closeness based on the similarity of their objectives.

3D clusters

Using the Sentence Bert model along with selected configurations, employees from Merck group were clustered on the basis of their objectives into 7 clusters. The 3D plot is shown in figure 8.26. This plot serves the same purpose as the 2D visualization but is able to retain more features.

LDA vis

LDA vis is a visualization for getting insight into the probability distribution within an LDA model. A snapshot of this visualization is shown in figure 8.27. It allows an in-depth analysis of the topic words belonging to each cluster. In the right side of the figure, it provides a description of cluster 7 in terms of the words that belong to it where the blue bar depicts the word's frequency in the whole corpus and the red bar depicts the word's frequency in the selected cluster.

8.3.2 Human Survey

A questionnaire [68] [69] [70] was created for the evaluation of the Topic Modeling Framework for Merck users who tested it. The questionnaire consists of 16 statements, divided into 8 evaluation aspects. The participant read the statement and expressed their agreement/disagreement with the statement on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree].

System Effectiveness

As shown in figure 8.28, most people who took the survey believe that the system is provides a smooth user experience and load any selected results quickly.

Effectiveness of Visualizations

As shown in figure 8.29 and 8.30, most people who took the survey liked the Employee cluster visualization in 2D and 3D. The feedback about word cloud as a visualization is ambiguous as more than 50% of the people liked it while the remaining didn't. This is depicted in figure 8.31. The response with respect the LDA vis is positive as most people think that is an effective form of visualization as shown in figure 8.32.

Results Quality

The survey response with respect to result quality is mostly positive. Most people think they the results provided by the system make sense to them as depicted in the histogram in figure 8.33. The survey response in figure 8.34 and 8.34 depict that most people who took the survey feel that the clusters formed by the framework are well-formulated and contain diverse topics. However, a few people had the opinion that results quality could be improved.

Privacy Concerns

As shown in figure 8.36, people who took the survey felt content about privacy aspect with respect to their data.

Effort to use the system

People who tested the system and filled out the survey think that the Topic Modeling framework is easy to use and navigate as shown in figure 8.37.

Layout Adequacy

As shown in figure 8.38, most people believe that the framework has an attractive appearance and contains right amount of explanations. However, only person felt the lack of explanations in framework which is depicted in figure 8.39.

Explainability and Transparency

As shown in figure 8.40, most people felt that the information provided along with the visualizations were sufficient. However, when asked about the extend to which the system educated them about the objectives of employees, there were mixed opinions. As depicted in figure 8.41, half of the people think that the system has helped them while the other half believes that it hasn't.

Use Intention

As shown in figure 8.42, most people claimed they would recommend the topic modeling framework to their colleagues. In terms of future usability, many people felt confident about using the framework in the future while some were not as confident, which is depicted in figure 8.43

Figure 8.6: Word Clouds when ELECTRA Embedding model is used as feature space for clustering before post-processing step with 10 clusters, frequency-based topic word retrieval approach on Job Description dataset.

(a) Cluster 1.

(b) Cluster 2.

(c) Cluster 3.

(d) Cluster 4.

(e) Cluster 5.

(f) Cluster 6.

(g) Cluster 7.

(h) Cluster 8.

(i) Cluster 9.

(j) Cluster 10.

Figure 8.7: Word Clouds when DistilBERT Embedding model is used as feature space for clustering before post-processing step with 10 clusters, frequency-based topic word retrieval approach on Job Description dataset.

(a) Cluster 1.

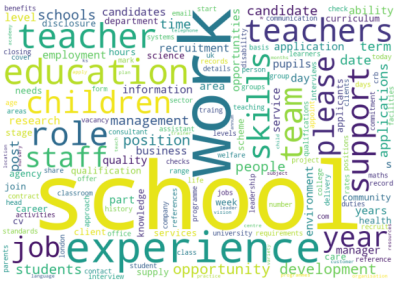
(b) Cluster 2.

(c) Cluster 3.

(d) Cluster 4.

(e) Cluster 5.

(f) Cluster 6.



(g) Cluster 7.

(h) Cluster 8.

(i) Cluster 9.

(j) Cluster 10.

Figure 8.8: Word Clouds when XLM Embedding model is used as feature space for clustering before post-processing step with 10 clusters, frequency-based topic word retrieval approach on Job Description dataset.

(a) Cluster 1.

(b) Cluster 2.

(c) Cluster 3.

(d) Cluster 4.

(e) Cluster 5.

(f) Cluster 6.

(g) Cluster 7.

(h) Cluster 8.

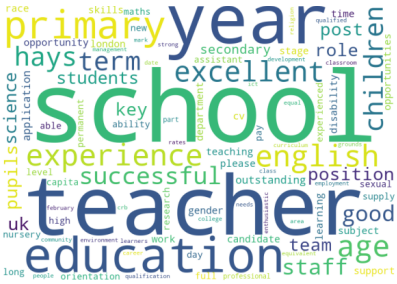
(i) Cluster 9.

(j) Cluster 10.

Figure 8.9: Word Clouds from an LDA (baseline) model with 10 clusters on Job Description Dataset.

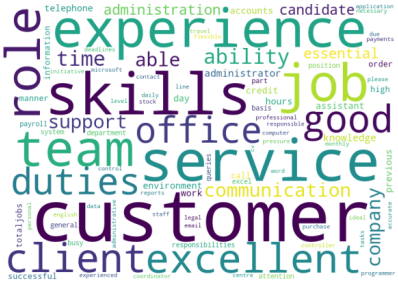
(a) Cluster 1.

(b) Cluster 2.



(c) Cluster 3.

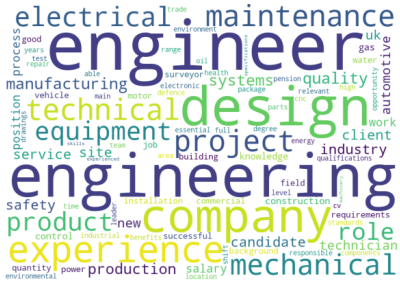
(d) Cluster 4.



(e) Cluster 5.

(f) Cluster 6.

(g) Cluster 7.



(h) Cluster 8.

(i) Cluster 9.

(j) Cluster 10.

Figure 8.10: Word Clouds from Feature combination of Sentence Bert Embedding model and LDA model for clustering before post-processing step with 10 clusters, frequency-based topic word retrieval approach on Job Description dataset.

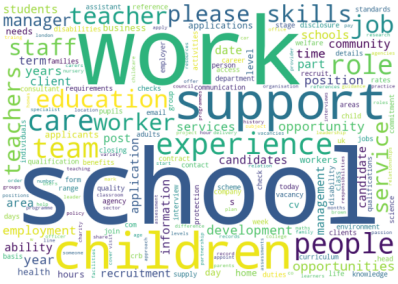
(a) Cluster 1.

(b) Cluster 2.

(c) Cluster 3.

(d) Cluster 4.

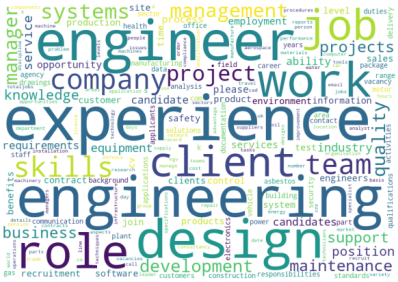
(e) Cluster 5.



(f) Cluster 6.

(g) Cluster 7.

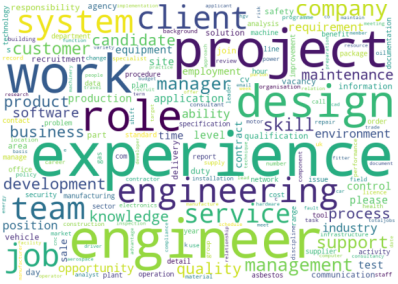
(h) Cluster 8.



(i) Cluster 9.

(j) Cluster 10.

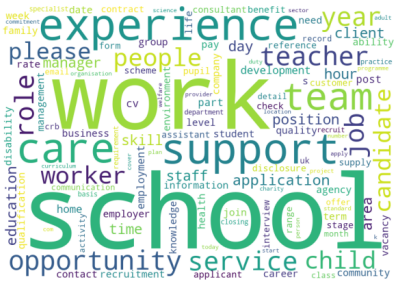
Figure 8.11: Word Clouds when latent representation of features from Sentence BERT Embedding model from an Autoencoder is used for clustering before post-processing step with 10 clusters, frequency-based topic word retrieval approach on Job Description dataset.



(a) Cluster 1.

(b) Cluster 2.

(c) Cluster 3.

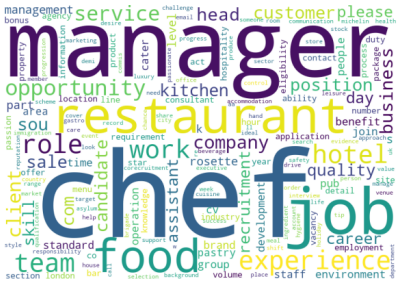


(d) Cluster 4.

(e) Cluster 5.

(f) Cluster 6.

(g) Cluster 7.

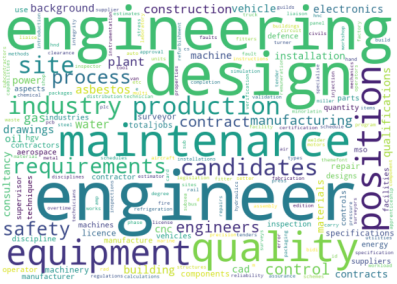


(h) Cluster 8.

(i) Cluster 9.

(j) Cluster 10.

Figure 8.12: Word Clouds when latent representation of features from Sentence BERT Embedding model + LDA features from an Autoencoder is used for clustering before post-processing step with 10 clusters, frequency-based topic word retrieval approach on Job Description dataset.



(a) Cluster 1.

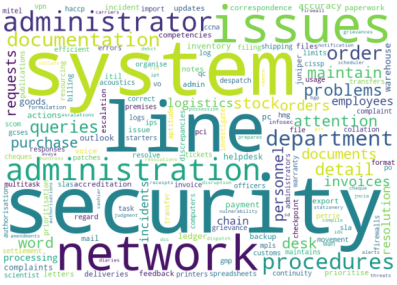
(b) Cluster 2.

(c) Cluster 3.

(d) Cluster 4.

(e) Cluster 5.

(f) Cluster 6.



(g) Cluster 7.

(h) Cluster 8.

(i) Cluster 9.

(j) Cluster 10.

Figure 8.13: Word Clouds with topic words obtained using self Attention of Bert Embedding model from huggingface library before post-processing step with 10 clusters on Job Description dataset.

(a) Cluster 1.

(b) Cluster 2.

(c) Cluster 3.

(d) Cluster 4.

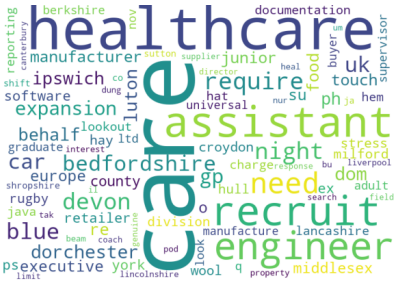
(e) Cluster 5.

(f) Cluster 6.

(g) Cluster 7.

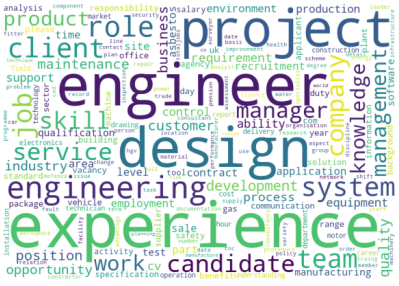
(h) Cluster 8.

(i) Cluster 9.



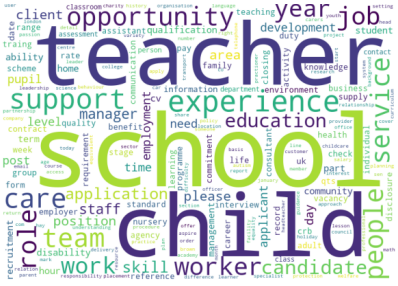
(j) Cluster 10.

Figure 8.14: Word Clouds with topic words obtained TF-IDF algorithm with feature vectors from Sentence BERT Embedding model for clustering before post-processing step with 10 clusters on Job Description dataset.

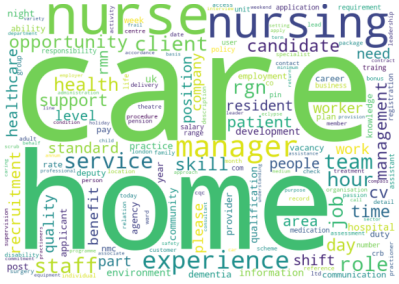


(a) Cluster 1.

(b) Cluster 2.



(c) Cluster 3.



(d) Cluster 4.

(e) Cluster 5.

(f) Cluster 6.

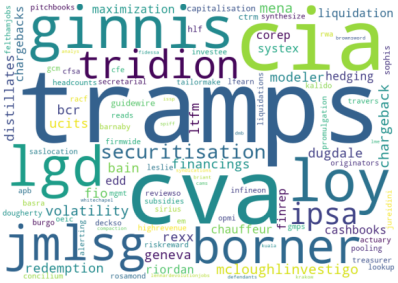
(g) Cluster 7.

(h) Cluster 8.

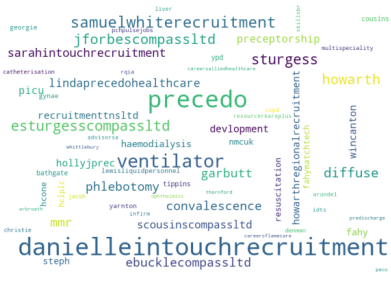
(i) Cluster 9.

(j) Cluster 10.

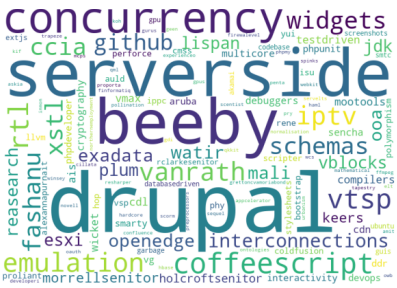
Figure 8.15: Word Clouds after post-processing using hard policy when Sentence BERT Embedding model is used as feature space for clustering with 10 clusters, frequency-based topic word retrieval approach on Job Description dataset.



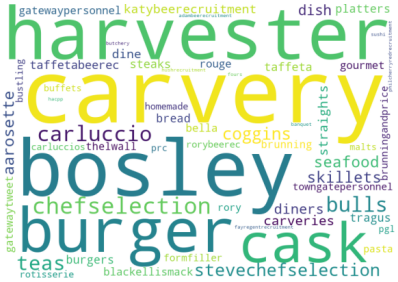
(a) Cluster 1.



(b) Cluster 2.



(c) Cluster 3.

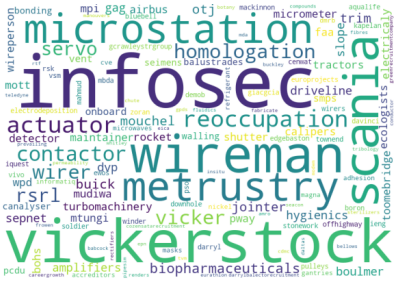


(d) Cluster 4.

(e) Cluster 5.

(f) Cluster 6.

(g) Cluster 7.



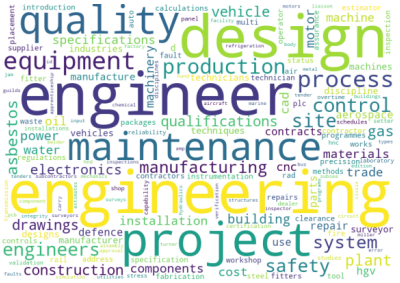
(h) Cluster 8.

(i) Cluster 9.

(j) Cluster 10.

Figure 8.16: Word Clouds after post-processing using soft policy when Sentence BERT Embedding model is used as feature space for clustering with 10 clusters, frequency-based topic word retrieval approach on Job Description dataset.

(a) Cluster 1.



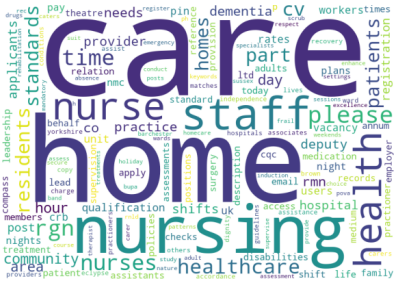
(b) Cluster 2.

(c) Cluster 3.

(d) Cluster 4.

(e) Cluster 5.

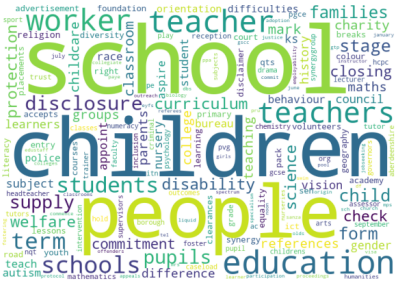
(f) Cluster 6.



(g) Cluster 7.

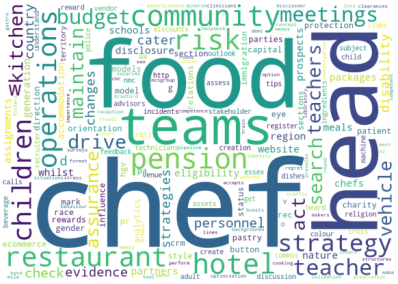
(h) Cluster 8.

(i) Cluster 9.



(j) Cluster 10.

Figure 8.17: Word Clouds after post-processing using soft policy when BERT Embedding model from huggingface library is used as feature space for clustering with 10 clusters, frequency-based topic word retrieval approach on Job Description dataset.

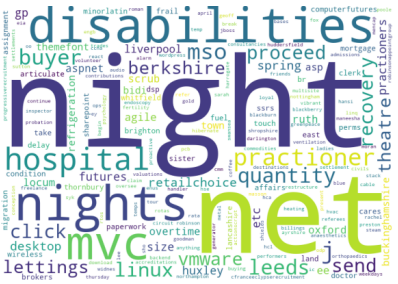


(a) Cluster 1.

(b) Cluster 2.

(c) Cluster 3.

(d) Cluster 4.



(e) Cluster 5.

(f) Cluster 6.

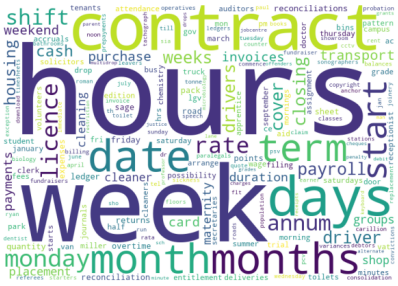
(g) Cluster 7.

(h) Cluster 8.

(i) Cluster 9.

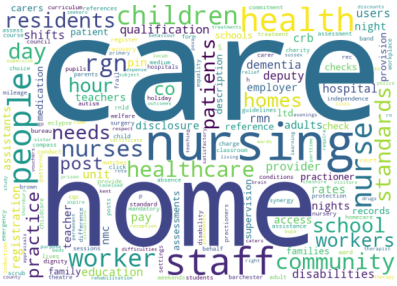
(j) Cluster 10.

Figure 8.18: Word Clouds with feature spaces from Sentence BERT Embedding model used for clustering with 5 clusters, after post-processing with soft policy, frequency-based topic word retrieval approach on Job Description dataset.



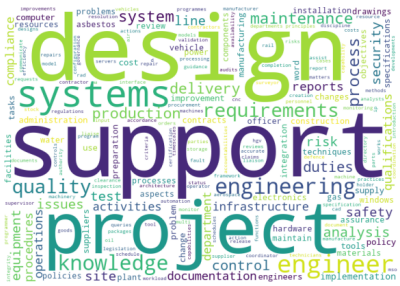
(a) Cluster 1.

(b) Cluster 2.



(c) Cluster 3.

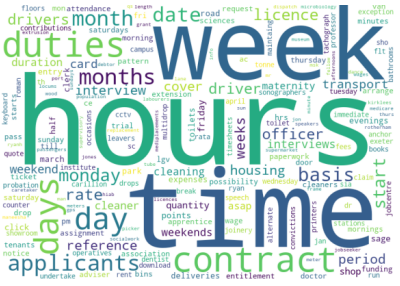
(d) Cluster 4.



(e) Cluster 5.

Figure 8.19: Word Clouds with feature spaces from Sentence BERT Embedding model used for clustering with 8 clusters, after post-processing with soft policy, frequency-based topic word retrieval approach on Job Description dataset.

(a) Cluster 1.

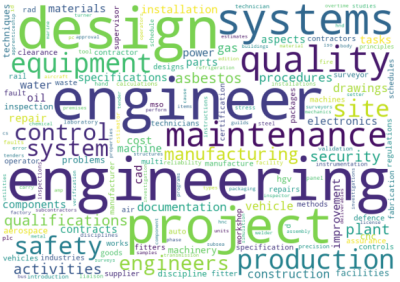


(b) Cluster 2.

(c) Cluster 3.

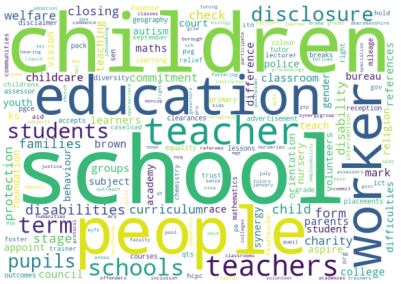
(d) Cluster 4.

(e) Cluster 5.



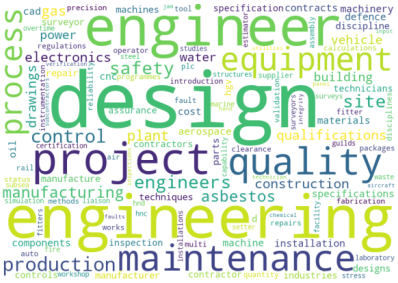
(f) Cluster 6.

(g) Cluster 7.



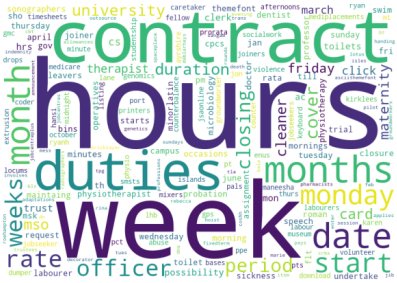
(h) Cluster 8.

Figure 8.20: Word Clouds with feature spaces from Sentence BERT Embedding model used for clustering with 12 clusters, after post-processing with soft policy, frequency-based topic word retrieval approach on Job Description dataset.



(a) Cluster 1.

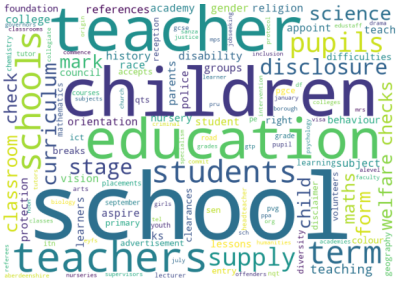
(b) Cluster 2.



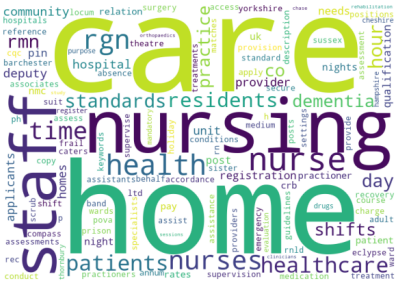
(c) Cluster 3.

(d) Cluster 4.

(e) Cluster 5.

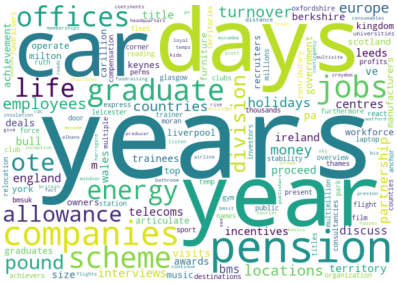


(f) Cluster 6.

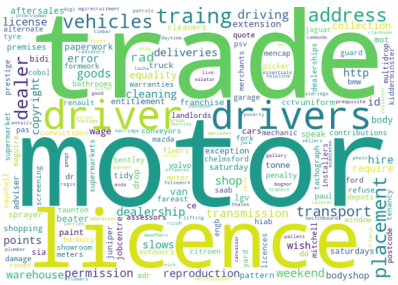


(g) Cluster 7.

(h) Cluster 8.



(i) Cluster 9.

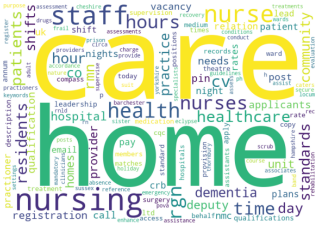


(j) Cluster 10.

(k) Cluster 11.

(l) Cluster 12.

Figure 8.21: Word Clouds with feature spaces from Sentence BERT Embedding model used for clustering with 14 clusters, after post-processing with soft policy, frequency-based topic word retrieval approach on Job Description dataset.



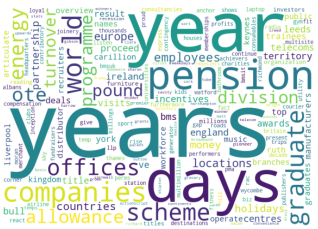
(a) Cluster 1.

(b) Cluster 2.

(c) Cluster 3.

(d) Cluster 4.

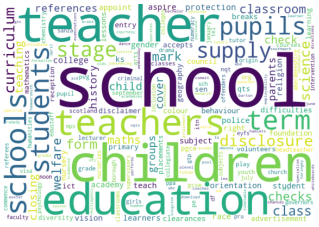
(e) Cluster 5.



(f) Cluster 6.

(g) Cluster 7.

(h) Cluster 8.

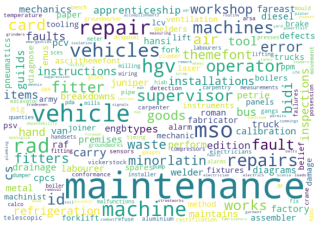


(i) Cluster 9.

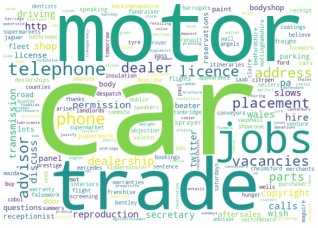
(j) Cluster 10.

(k) Cluster 11.

(l) Cluster 12.



(m) Cluster 13.



(n) Cluster 14.

Figure 8.22: Word Clouds with feature spaces from Sentence BERT Embedding model used for clustering with 12 clusters, after post-processing with soft policy, frequency-based topic word retrieval approach and N-gram = 2 on Job Description dataset.

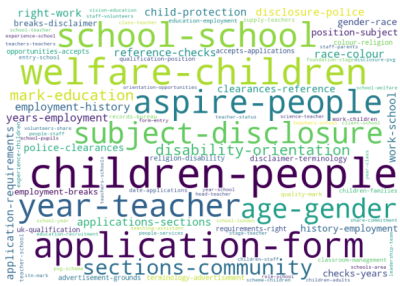
(a) Cluster 1.

(b) Cluster 2.

(c) Cluster 3.

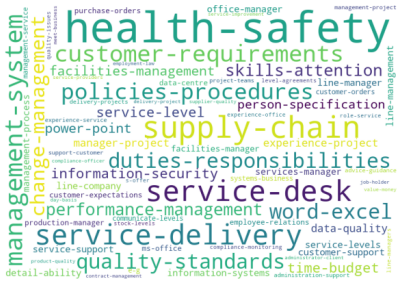
(d) Cluster 4.

(e) Cluster 5.



(f) Cluster 6.

(g) Cluster 7.



(h) Cluster 8.

(i) Cluster 9.

(j) Cluster 10.

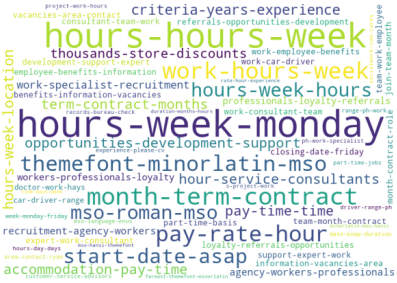
(k) Cluster 11.

(l) Cluster 12.

Figure 8.23: Word Clouds with feature spaces from Sentence BERT Embedding model used for clustering with 12 clusters, after post-processing with soft policy, frequency-based topic word retrieval approach and N-gram = 3 on Job Description dataset.

(a) Cluster 1.

(b) Cluster 2.



(c) Cluster 3.

(d) Cluster 4.

(e) Cluster 5.

(f) Cluster 6.

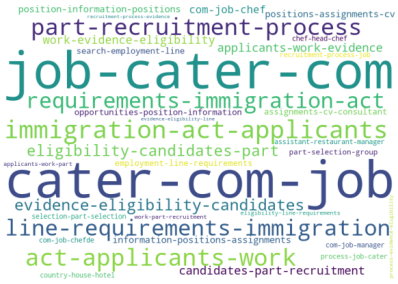
(g) Cluster 7.

(h) Cluster 8.

(i) Cluster 9.

(j) Cluster 10.

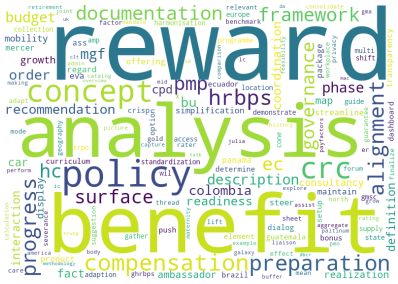
(k) Cluster 11.



(l) Cluster 12.

Figure 8.24: Word Clouds with feature spaces from Sentence BERT Embedding model used for clustering with 7 clusters, after post-processing with soft policy, frequency-based topic word retrieval approach and N-gram = 1 on Employee Objective dataset.

(a) Cluster 1.



(b) Cluster 2.

(c) Cluster 3.

(d) Cluster 4.

(e) Cluster 5.

(f) Cluster 6.

(g) Cluster 7.

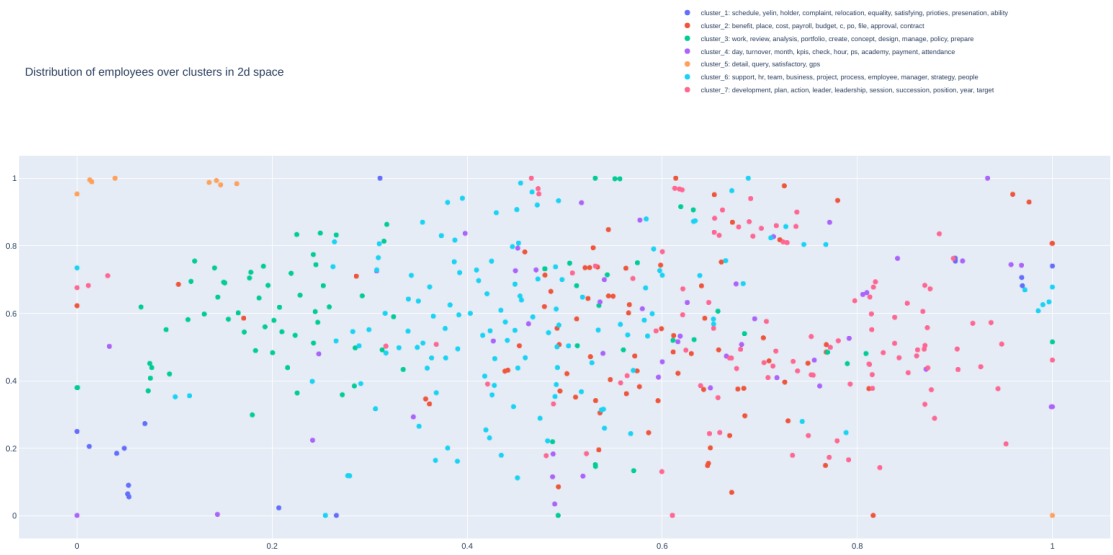


Figure 8.25: Clustering of Merck Group Employees in a 2D space with feature spaces from Sentence BERT Embedding model used for clustering with 7 clusters, after post-processing with soft policy, frequency-based topic word retrieval approach and N-gram = 1 on Employee Objective dataset.

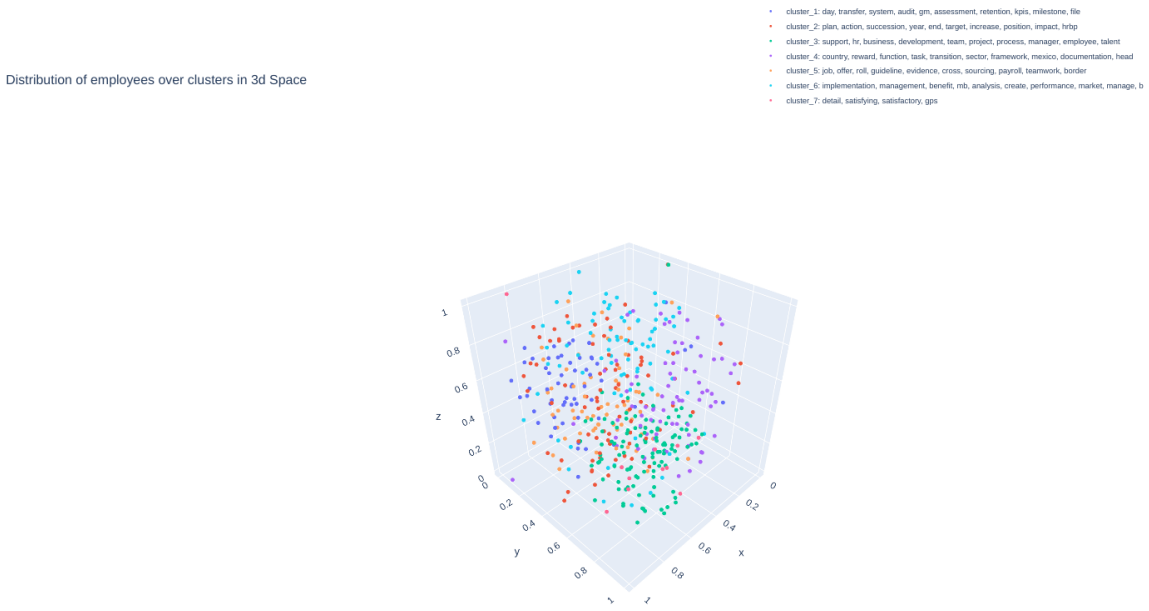


Figure 8.26: Clustering of Merck Group Employees in a 3D space with feature spaces from Sentence BERT Embedding model used for clustering with 7 clusters, after post-processing with soft policy, frequency-based topic word retrieval approach and N-gram = 1 on Employee Objective dataset.

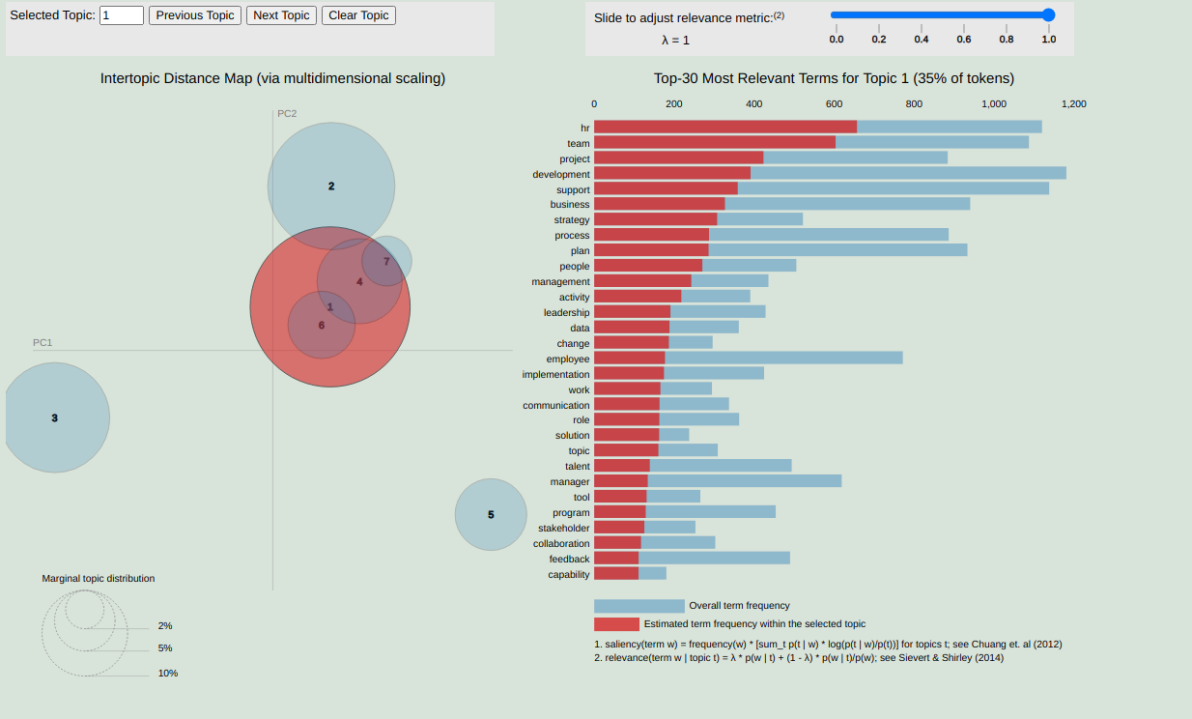


Figure 8.27: A visualization of the LDA model on Employee Objective dataset where employees are grouped in 7 clusters. Left side of image shows distance between clusters and the right side of image shows distribution of topic words in each cluster.

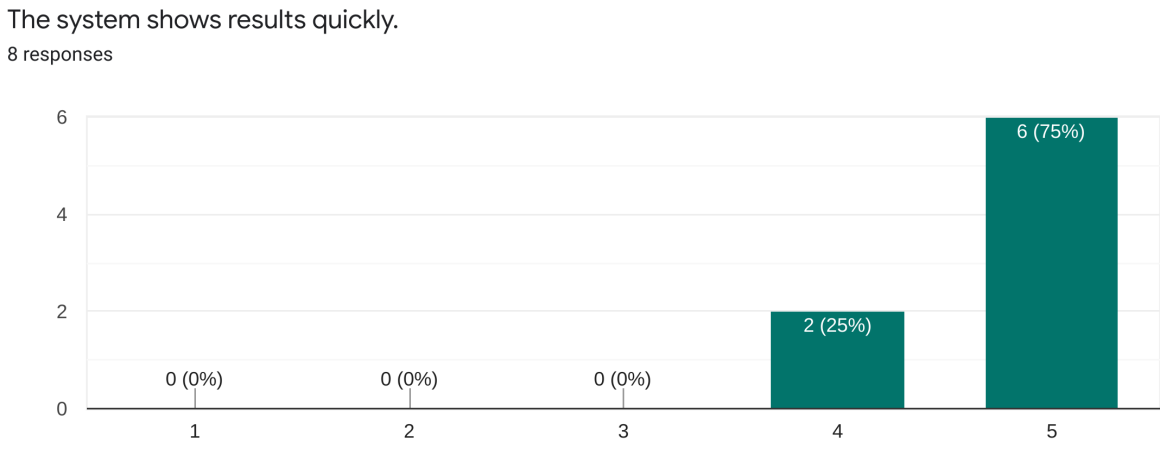


Figure 8.28: A histogram of feedback from Merck Group on System Effectiveness. The participant read the statement and expressed their agreement/disagreement with the statement on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree]

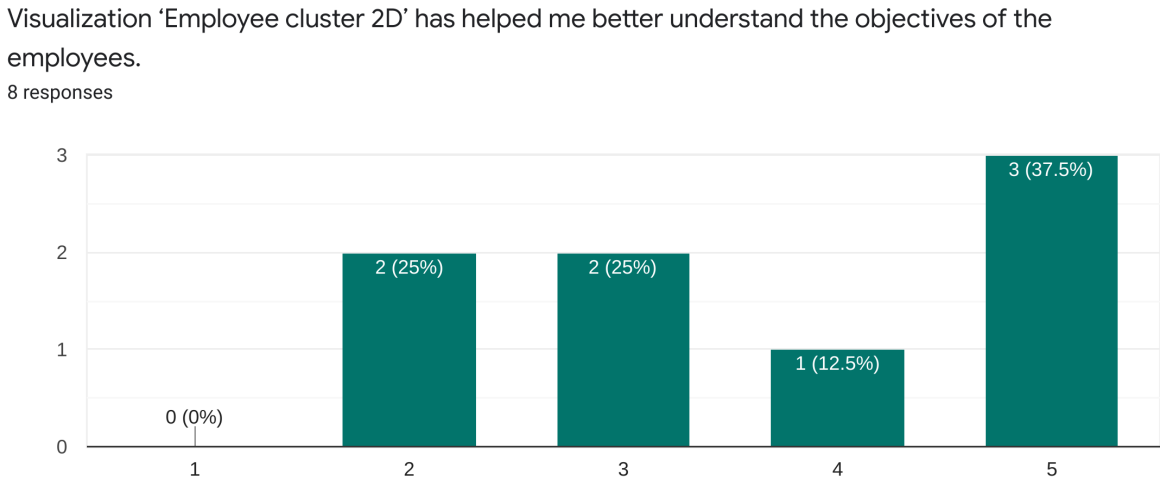


Figure 8.29: A histogram of feedback from Merck Group on Employee clusters in 2D. The participant read the statement and expressed their agreement/disagreement with the statement on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree]

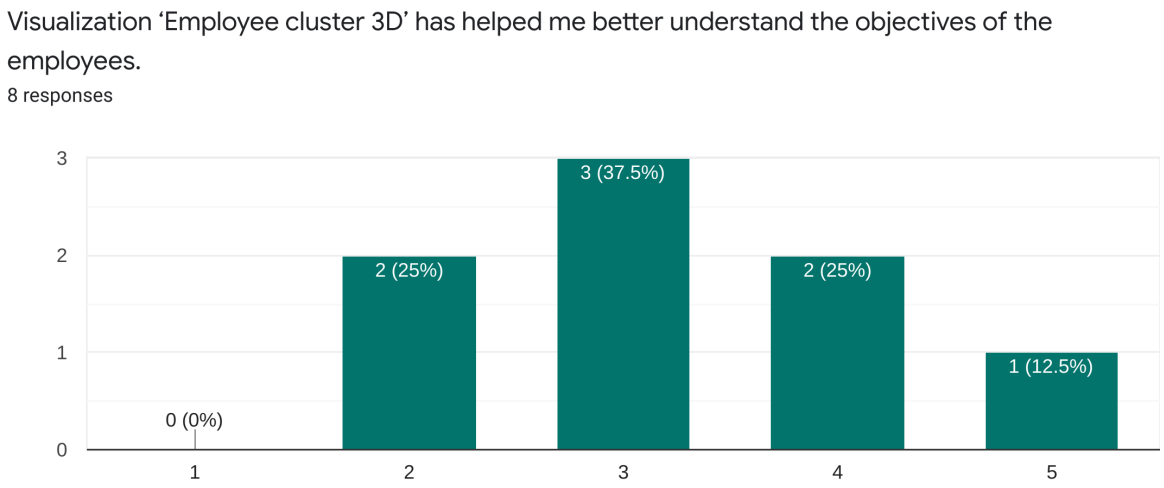


Figure 8.30: A histogram of feedback from Merck Group on Employee clusters in 3D. The participant read the statement and expressed their agreement/disagreement with the statement on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree]

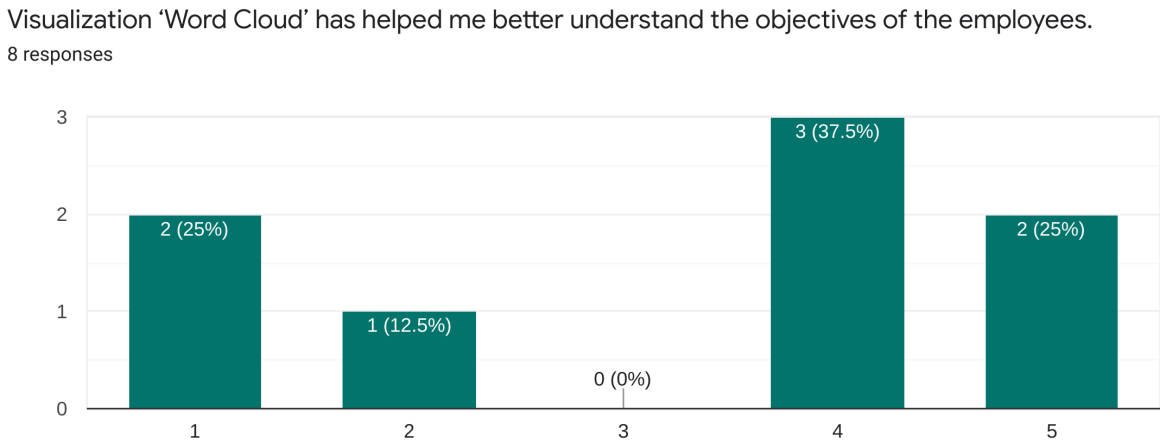


Figure 8.31: A histogram of feedback from Merck Group on Word clouds visualization. The participant read the statement and expressed their agreement/disagreement with the statement on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree]

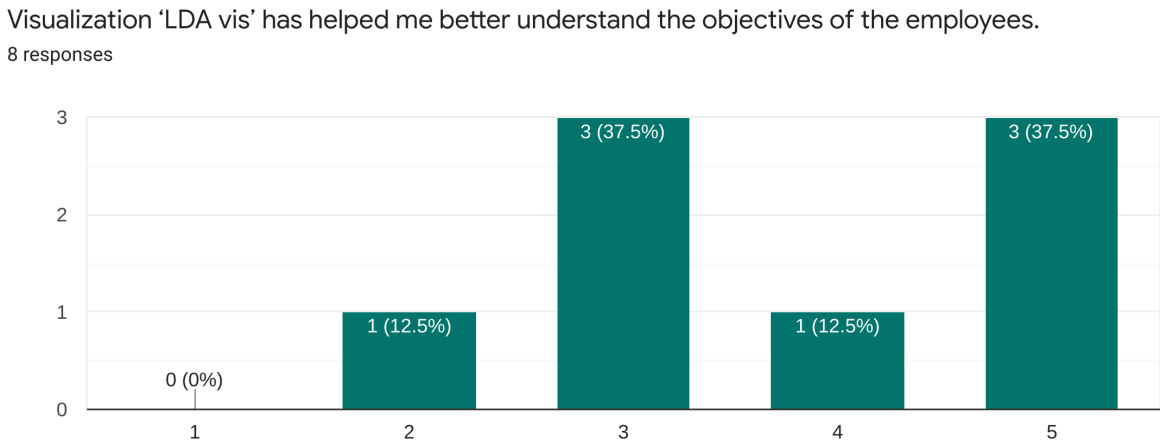


Figure 8.32: A histogram of feedback from Merck Group on 'LDA vis' as a visualization. The participant read the statement and expressed their agreement/disagreement with the statement on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree]

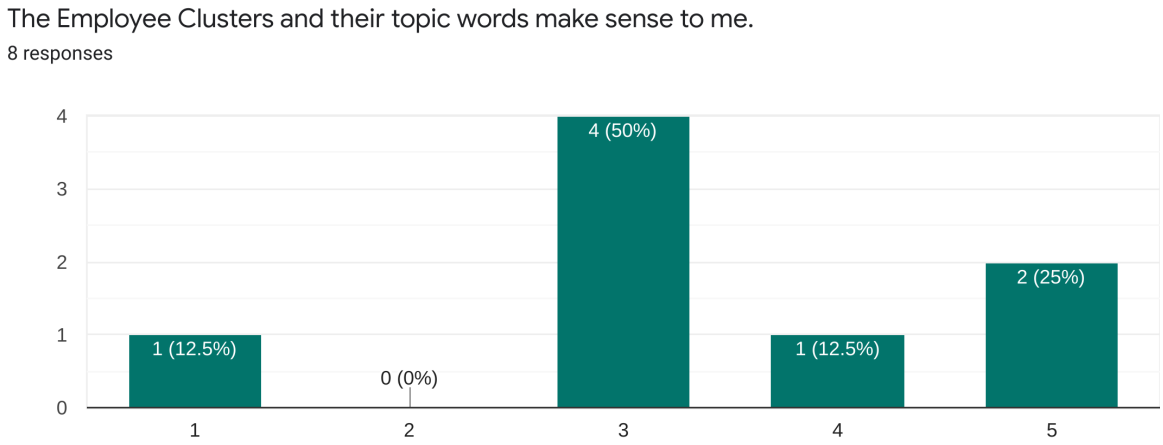


Figure 8.33: A histogram of feedback from Merck Group on the sensibility of results. The participant read the statement and expressed their agreement/disagreement with the statement on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree]

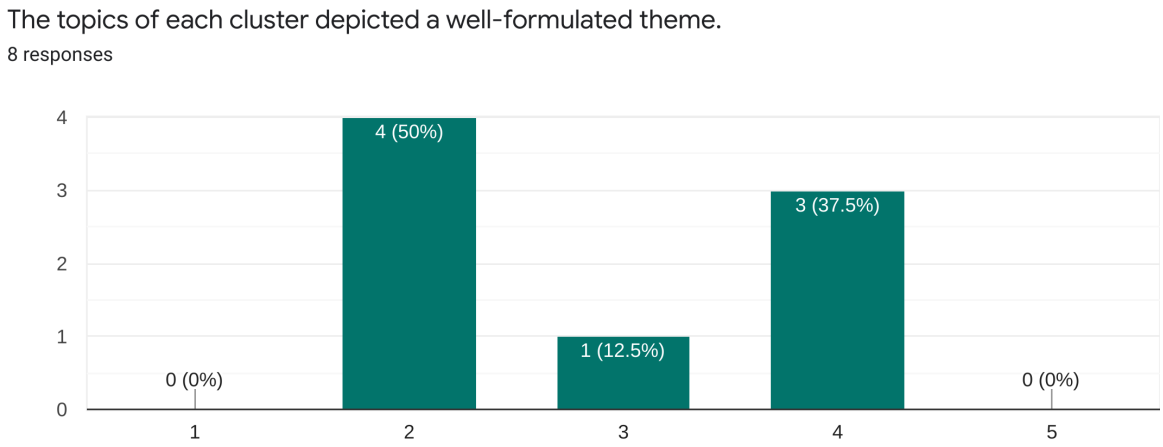


Figure 8.34: A histogram of feedback from Merck Group on well-formulated themes of clusters. The participant read the statement and expressed their agreement/disagreement with the statement on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree]

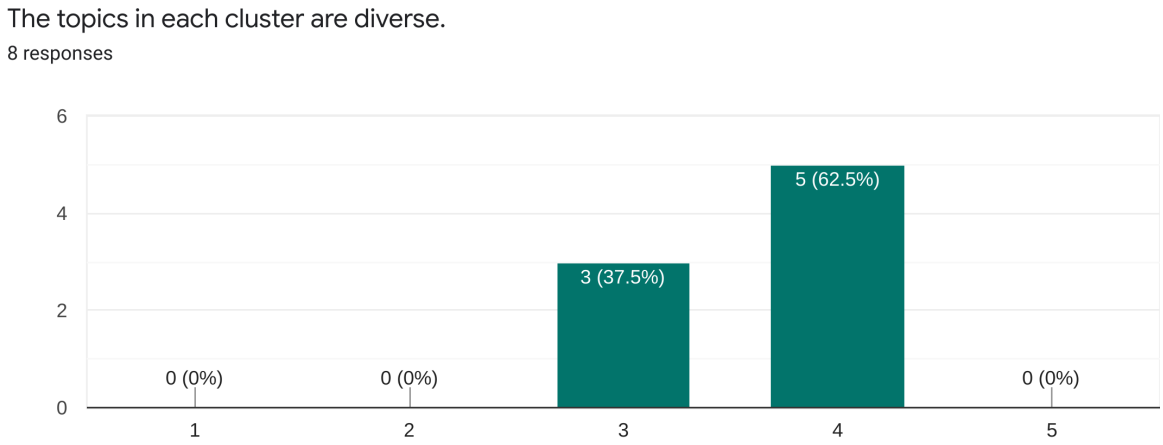


Figure 8.35: A histogram of feedback from Merck Group on cluster diversity. The participant read the statement and expressed their agreement/disagreement with the statement on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree]

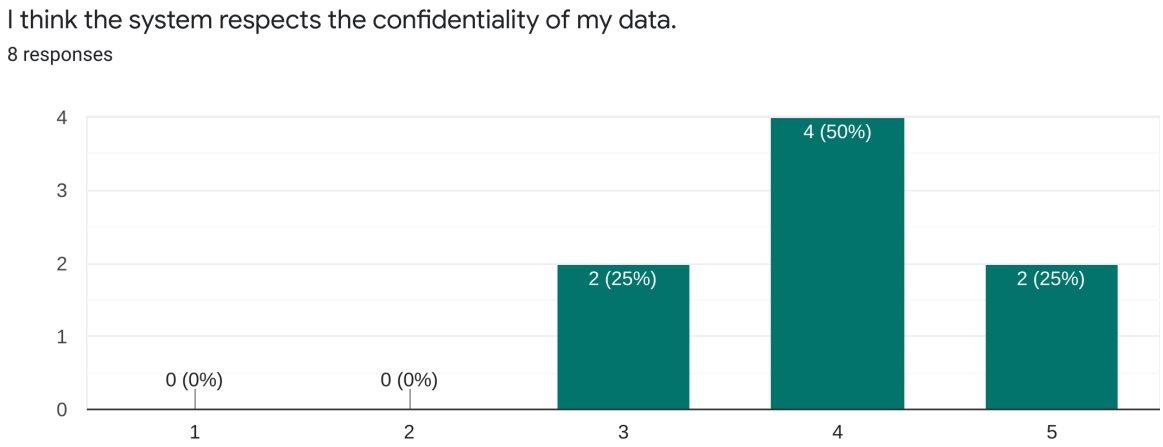


Figure 8.36: A histogram of feedback from Merck Group on data privacy concerns. The participant read the statement and expressed their agreement/disagreement with the statement on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree]

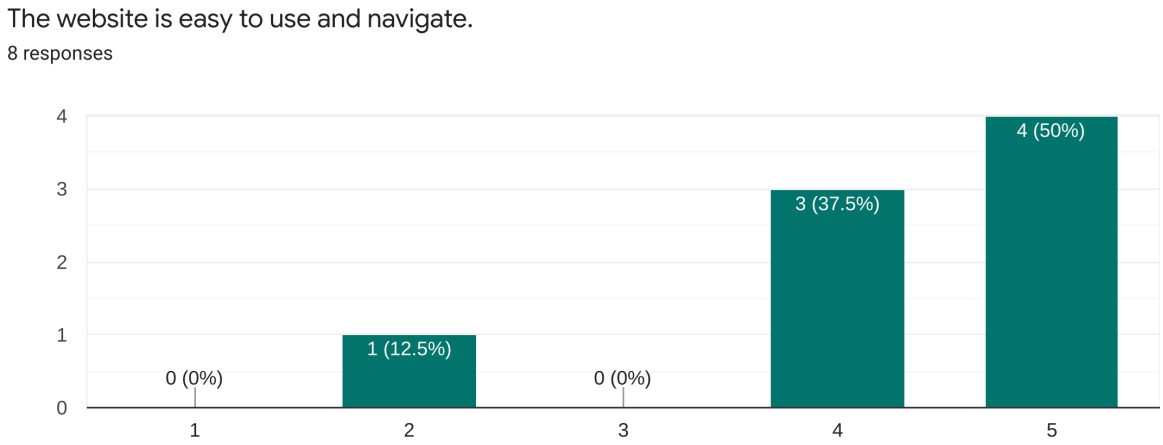


Figure 8.37: A histogram of feedback from Merck Group on efforts to use the system. The participant read the statement and expressed their agreement/disagreement with the statement on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree]

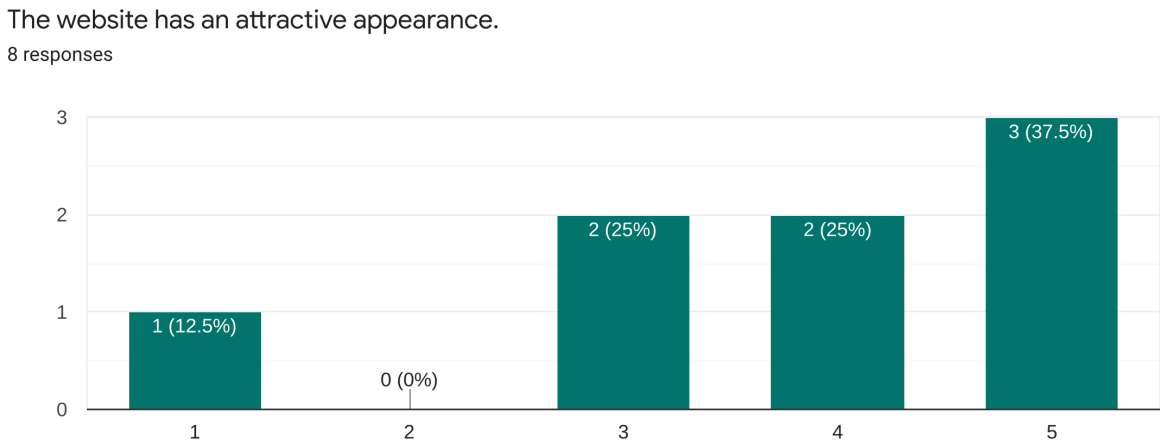


Figure 8.38: A histogram of feedback from Merck Group on system appearance. The participant read the statement and expressed their agreement/disagreement with the statement on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree]e

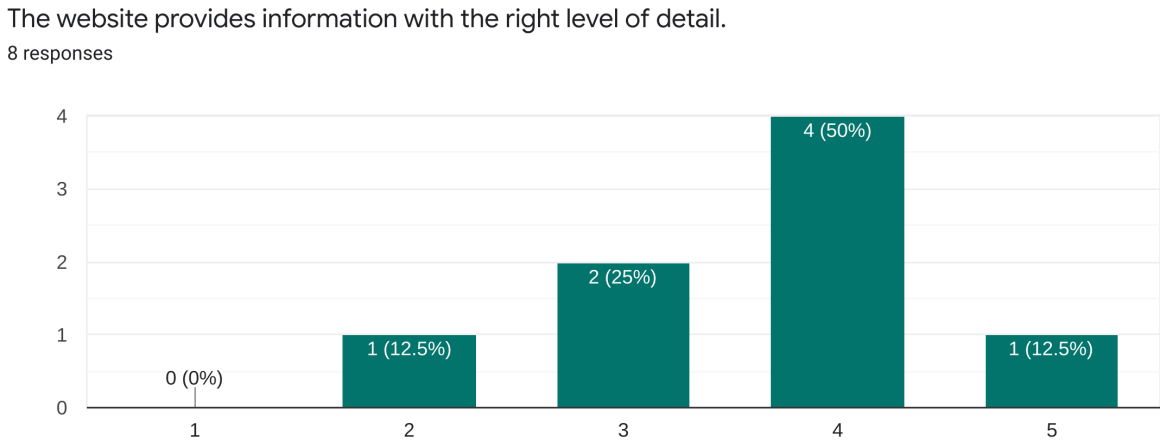


Figure 8.39: A histogram of feedback from Merck Group on detail of information provided. The participant read the statement and expressed their agreement/disagreement with the statement on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree]

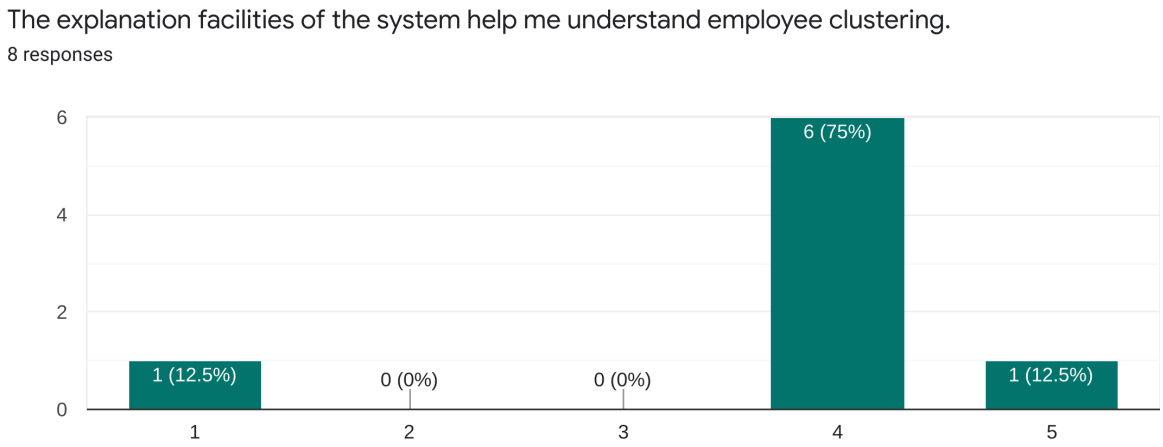


Figure 8.40: A histogram of feedback from Merck Group on explanations for visualizations. The participant read the statement and expressed their agreement/disagreement with the statement on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree]

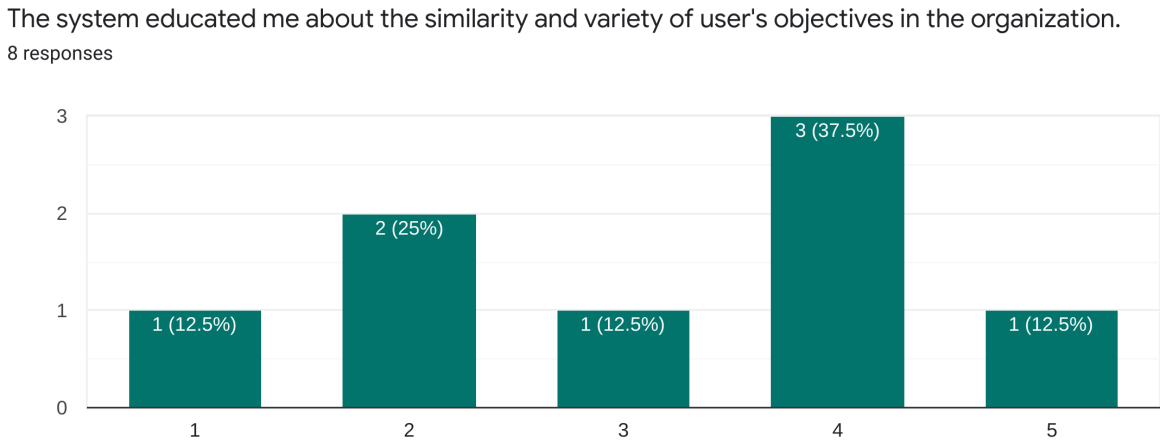


Figure 8.41: A histogram of feedback from Merck Group on the effectiveness of the system. The participant read the statement and expressed their agreement/disagreement with the statement on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree]

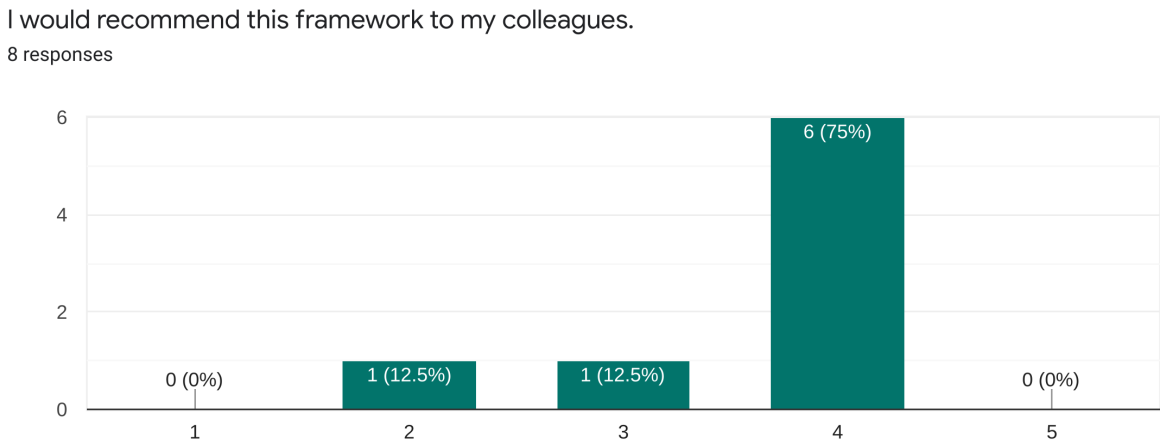


Figure 8.42: A histogram of feedback from Merck Group on recommendation to colleagues. The participant read the statement and expressed their agreement/disagreement with the statement on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree]

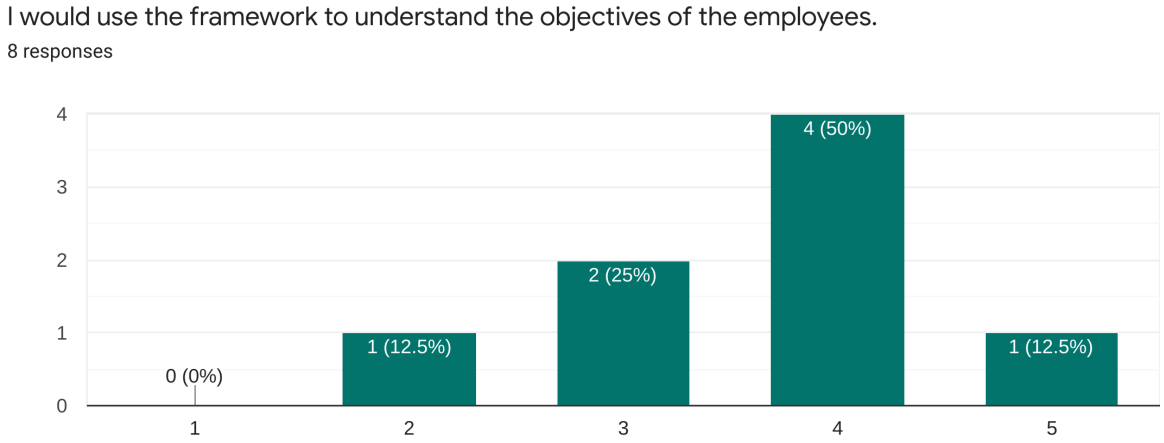


Figure 8.43: A histogram of feedback from Merck Group on usage of framework in future. The participant read the statement and expressed their agreement/disagreement with the statement on a 1-5 scale: [Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree]

9 Conclusion

This study focused on performing Topic Modeling on a Job Description Dataset and Employee Objective Dataset. Experiments were conducted on Job Description dataset to find an appropriate feature space for clustering and topic words retrieval separately. To check the clustering results, word clouds were generated using feature vectors obtained from 7 Embedding models with other configurations set as default. these models include Sentence BERT [4], Sentence RoBERTa [61] and Word Embedding models including Sentence DistilBERT [62], BERT [3], XLNET [5], XLM [59] and Electra [60]. Through experiments it was concluded that features obtained from Sentence Transformer models provide better results. Among the three Sentence transformers, Sentence BERT provided the most coherent clusters. One theory behind the poor performance of Embedding models such as BERT and XLNET could be that token level representation is not well suited for clustering tasks. Whereas, the Sentence transformers provide better clustering as feature vectors provide a better representation of the text as a whole. On further experimentation, it was found that using Word Embedding for Topic Modeling can indeed produce better results than LDA. Furthermore, it was also found that using Word Embedding features in tandem with LDA or using the latent representation from an Autoencoder provided the same results as using the Embedding model features alone.

To observe the results of Topic Modeling, word clouds were created for three techniques including a frequency-based techniques, TF-IDF and a Transformer Attention approach. It was found that Frequency-based approach and TF-IDF provided most coherent topic words for clusters whereas the self Attention of a Transformer model didn't give meaningful topics. In terms of N-gram, 1-gram approach had the most coherent clusters and the 2-gram approach provided a bit noisy but still coherent cluster. The results of the 3-gram approach failed to retain any of the cluster theme. In the end, the extracted topic words had a lot of redundancies within cluster. An additional post processing step was used to remove these redundant topic words. The output after this step was more coherent topic words for cluster. In general, it was observed that the analytical scores used for analysis of results contradicted with the Human evaluation of Word Clouds. This is an indicator that these scores are not suitable for a Topic Modeling task such as this one.

In terms of Topic Modeling results, the Job description dataset provided coherent clusters and topic words. This confirms that the model configurations are correct. However, the results on Employee Objective dataset are not as good. Employee clusters formed and their topic words are not as coherent as the other Job clusters. This is because the employee objective dataset doesn't contain any meaningful data and is also very small in size.

The Topic Modeling Framework discussed in chapter 4 was reviewed by Merck Group. Several people gave their feedback by filling out a survey. The general impression was positive. The survey concluded that the framework provided a smooth user experience and the users

most appreciated the Graphical user interface as it was easy to use and navigate. In terms of visualisations, 'LDA vis' and 2D representation of employees were most well received. The users felt that the framework provided appropriate amount of information regarding the visualization. However, many people felt that the topic words in the clusters weren't as coherent and informative which is due to poor quality of data. The survey concluded the framework abides by the privacy laws. In the end, Most people responded in positive when asked if they would recommend the framework to other colleagues and also using it in the future to understand the objectives of employees.

10 Future Work

Throughout the thesis work, experiments were conducted to find the best configuration of techniques for Topic Modeling. Even though the coherent clusters were obtained for the Job Description dataset, the clusters obtained using Employee objective dataset were not as coherent. The current dataset obtained from Merck Group contains many redundant and noisy data points. The Topic Modeling results for Employee objective dataset could be improved by obtaining a more structured and bigger dataset that contains employees from multiple departments. Number of experiments were reduced to a certain limit due the time limit in this thesis. Further research could include experimenting with different configurations of deep learning model and perhaps also fine-tuning using the masking approach followed by state-of-the-art models such as BERT [3] and XLNET [5].

The GUI of the Topic Modeling framework is created using flask along with some basic html code. The framework could be revised by using a front-end tool such as React and making it more dynamic and user-friendly. Similarly, more work could be done on the analytic section of the framework. Currently, it only focuses on clustering the employees and retrieving topic words. This could be extended by including analytic functionalities such the evaluation of employee objectives with the goal of the companies. Another option is to include a functionality to track the objectives of employees over the years and visualize how they have evolved.

List of Figures

List of Tables

Bibliography

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent Dirichlet Allocation". In: *J. Mach. Learn. Res.* 3.null (Mar. 2003), pp. 993–1022. ISSN: 1532-4435.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[4] N. Reimers and I. Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 2019. arXiv: 1908.10084 [cs.CL].

[5] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. 2020. arXiv: 1906.08237 [cs.CL].

[6] J. Wilson. *Essentials of Business Research: A Guide to Doing Your Research Project*. SAGE Publications, p.7, 2010.

[7] D. Khurana, A. Koli, K. Khatter, and S. Singh. "Natural Language Processing: State of The Art, Current Trends and Challenges". In: (Aug. 2017).

[8] A. Torfi, R. Shirvani, Y. Keneshloo, N. Tavvaf, and E. Fox. *Natural Language Processing Advancements By Deep Learning: A Survey*. Mar. 2020.

[9] S. Sanampudi and K. G.Vijaya. "Temporal Reasoning in Natural Language Processing: A Survey". In: *International Journal of Computer Applications* 1 (Feb. 2010). DOI: 10.5120/100-209.

[10] Z. Khalifehlou. "Analysis and evaluation of unstructured data: Text mining versus natural language processing". In: Nov. 2011, pp. 1–4. DOI: 10.1109/ICAICT.2011.6111017.

[11] R. Habibpour and K. Khalilpour. "A new hybrid k-means and K-nearest-neighbor algorithms for text document clustering". In: *International Journal of Academic Research* 6 (May 2014), pp. 79–84. DOI: 10.7813/2075-4124.2014/6-3/A.12.

[12] S. Rose, D. Engel, N. Cramer, and W. Cowley. "Automatic Keyword Extraction from Individual Documents". In: Mar. 2010, pp. 1–20. ISBN: 9780470689646. DOI: 10.1002/9780470689646.ch1.

[13] H. M. Hasan, F. Sanyal, D. Chaki, and M. Ali. "An empirical study of important keyword extraction techniques from documents". In: Dec. 2017. DOI: 10.1109/ICISIM.2017.8122154.

[14] M. Timonen, T. Toivanen, M. Kasari, Y. Teng, C. Cheng, and L. He. "Keyword Extraction from Short Documents Using Three Levels of Word Evaluation". In: vol. 415. Jan. 2013, pp. 130–146. ISBN: 9783642541049. DOI: 10.1007/978-3-642-54105-6_9.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[16] J. Li, Q. Fan, and K. Zhang. "Keyword extraction based on tf/idf for Chinese news document". In: *Wuhan University Journal of Natural Sciences* 12 (Sept. 2007), pp. 917–921. DOI: 10.1007/s11859-007-0038-4.

[17] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis. "Deep Learning for Computer Vision: A Brief Review". In: *Computational Intelligence and Neuroscience* 2018 (Feb. 2018), p. 7068349. ISSN: 1687-5265. DOI: 10.1155/2018/7068349. URL: https://doi.org/10.1155/2018/7068349.

[18] J. Walsh, N. O' Mahony, S. Campbell, A. Carvalho, L. Krpalkova, G. Velasco-Hernandez, S. Harapanahalli, and D. Riordan. "Deep Learning vs. Traditional Computer Vision". In: Apr. 2019. ISBN: 978-981-13-6209-5. DOI: 10.1007/978-3-030-17795-9_10.

[19] D. W. Otter, J. R. Medina, and J. K. Kalita. *A Survey of the Usages of Deep Learning in Natural Language Processing*. 2019. arXiv: 1807.10854 [cs.CL].

[20] A. Torfi, R. A. Shirvani, Y. Keneshloo, N. Tavaf, and E. A. Fox. *Natural Language Processing Advancements By Deep Learning: A Survey*. 2020. arXiv: 2003.01200 [cs.CL].

[21] R. Zemouri, N. Zerhouni, and D. Racoceanu. "Deep Learning in the Biomedical Applications: Recent and Future Status". In: *Applied Sciences* 9 (Apr. 2019), p. 1526. DOI: 10.3390/app9081526.

[22] C. Cao, F. Liu, H. Tan, D. Song, W. Shu, W. Li, Y. Zhou, X. Bo, and Z. Xie. "Deep Learning and Its Applications in Biomedicine". In: *Genomics, Proteomics & Bioinformatics* 16 (Mar. 2018). DOI: 10.1016/j.gpb.2017.07.003.

[23] C. Cao, F. Liu, H. Tan, D. Song, W. Shu, W. Li, Y. Zhou, X. Bo, and Z. Xie. "Deep Learning and Its Applications in Biomedicine". In: *Genomics, Proteomics & Bioinformatics* 16.1 (2018), pp. 17–32. ISSN: 1672-0229. DOI: https://doi.org/10.1016/j.gpb.2017.07.003. URL: http://www.sciencedirect.com/science/article/pii/S1672022918300020.

[24] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. 2018. arXiv: 1811.03378 [cs.LG].

[25] J. Han and C. Moraga. "The Influence of the Sigmoid Function Parameters on the Speed of Backpropagation Learning". In: *Proceedings of the International Workshop on Artificial Neural Networks: From Natural to Artificial Neural Computation*. IWANN '96. Berlin, Heidelberg: Springer-Verlag, 1995, pp. 195–201. ISBN: 3540594973.

[26] A. F. Agarap. *Deep Learning using Rectified Linear Units (ReLU)*. 2019. arXiv: 1803.08375 [cs.NE].

[27] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin. "Backpropagation: The Basic Theory". In: *Backpropagation: Theory, Architectures, and Applications*. USA: L. Erlbaum Associates Inc., 1995, pp. 1–34. ISBN: 0805812598.

[28] S. Ruder. *An overview of gradient descent optimization algorithms.* cite arxiv:1609.04747Comment: Added derivations of AdaMax and Nadam. 2016. URL: http://arxiv.org/abs/1609.04747.

[29] D. H. Ballard. "Modular Learning in Neural Networks". In: *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*. AAAI'87. Seattle, Washington: AAAI Press, 1987, pp. 279–284. ISBN: 0934613427.

[30] P. Baldi. "Autoencoders, Unsupervised Learning and Deep Architectures". In: *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27*. UTLW'11. Washington, USA: JMLR.org, 2011, pp. 37–50.

[31] X. Li, T. Zhang, X. Zhao, and Z. Yi. "Guided autoencoder for dimensionality reduction of pedestrian features". In: *Applied Intelligence* 50.12 (Dec. 2020), pp. 4557–4567. ISSN: 1573-7497. DOI: 10.1007/s10489-020-01813-1. URL: https://doi.org/10.1007/s10489-020-01813-1.

[32] L. C. Jain and L. R. Medsker. *Recurrent Neural Networks: Design and Applications*. 1st. USA: CRC Press, Inc., 1999. ISBN: 0849371813.

[33] S. Hochreiter and J. Schmidhuber. *LONG SHORT-TERM MEMORY*. 1997.

[34] J. Camacho-Collados and M. T. Pilehvar. "From Word to Sense Embeddings: A Survey on Vector Representations of Meaning". In: *J. Artif. Int. Res.* 63.1 (Sept. 2018), pp. 743–788. ISSN: 1076-9757. DOI: 10.1613/jair.1.11259. URL: https://doi.org/10.1613/jair.1.11259.

[35] F. Almeida and G. Xexéo. *Word Embeddings: A Survey*. 2019. arXiv: 1901.09069 [cs.CL].

[36] X. Rong. *word2vec Parameter Learning Explained*. 2016. arXiv: 1411.2738 [cs.CL].

[37] P. Rai and S. Shubha. "A Survey of Clustering Techniques". In: *International Journal of Computer Applications* 7 (Oct. 2010). DOI: 10.5120/1326-1808.

[38] S. Bano and N. Khan. "A Survey of Data Clustering Methods". In: *International Journal of Advanced Science and Technology* 113 (Apr. 2018). DOI: 10.14257/ijast.2018.113.14.

[39] D. Xu and Y. Tian. "A Comprehensive Survey of Clustering Algorithms". In: *Annals of Data Science* 2.2 (June 2015), pp. 165–193. ISSN: 2198-5812. DOI: 10.1007/s40745-015-0040-1. URL: https://doi.org/10.1007/s40745-015-0040-1.

[40] X. Jin and J. Han. "K-Means Clustering". In: *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010, pp. 563–564. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_425. URL: https://doi.org/10.1007/978-0-387-30164-8_425.

[41] C. Mulunda, P. Wagacha, and L. Muchemi. "Review of Trends in Topic Modeling Techniques, Tools, Inference Algorithms and Applications". In: Nov. 2018, pp. 28–37. DOI: 10.1109/ISCMI.2018.8703231.

[42] P. Kherwa and P. Bansal. "Topic Modeling: A Comprehensive Review". In: *ICST Transactions on Scalable Information Systems* 7 (July 2018), p. 159623. DOI: 10.4108/eai.13-7-2018.159623.

[43] C. E. Moody. *Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec*. 2016. arXiv: 1605.02019 [cs.CL].

[44] A. B. Dieng, F. J. R. Ruiz, and D. M. Blei. *Topic Modeling in Embedding Spaces*. 2019. arXiv: 1907.04907 [cs.IR].

[45] L. Y. Miao and P. Blunsom. *Neural variational inference for text processing*. 2016.

[46] D. M. S. Z. Ananya Sarker S.M. Shamim and M. M. Rahman. *Employee's Performance Analysis and Prediction using K-Means Clustering & Decision Tree Algorithm*. 2018.

[47] M. Nasr, E. Shaaban, and A. Samir. "A proposed Model for Predicting Employees' Performance Using Data Mining Techniques: Egyptian Case Study". In: *International Journal of Computer Science and Information Security* 17 (Jan. 2019), pp. 31–40.

[48] E. Faliagka, K. Ramantas, A. Tsakalidis, and G. Tzimas. "Application of Machine Learning Algorithms to an online Recruitment System". In: Jan. 2012.

[49] Y. Zhao, M. Hryniewicki, F. Cheng, B. Fu, and X. Zhu. "Employee Turnover Prediction with Machine Learning: A Reliable Approach". In: Jan. 2019, pp. 737–758. ISBN: 978-3-030-01056-0. DOI: 10.1007/978-3-030-01057-7_56.

[50] W. A. Belson. "Matching and Prediction on the Principle of Biological Classification". In: *Journal of the Royal Statistical Society Series C* 8.2 (June 1959), pp. 65–75. DOI: 10.2307/2985543. URL: https://ideas.repec.org/a/bla/jorssc/v8y1959i2p65-75.html.

[51] T. Evgeniou and M. Pontil. "Support Vector Machines: Theory and Applications". In: vol. 2049. Jan. 2001, pp. 249–257. DOI: 10.1007/3-540-44673-7_12.

[52] T. Chen and C. Guestrin. "XGBoost". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Aug. 2016). DOI: 10.1145/2939672.2939785. URL: http://dx.doi.org/10.1145/2939672.2939785.

[53] R. DuPlain. *Flask Web Development*. Aug. 2013. ISBN: 1782169628.

[54] C. Sievert and K. Shirley. "LDAvis: A method for visualizing and interpreting topics". In: June 2014. DOI: 10.13140/2.1.1394.3043.

[55] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 [cs.CL].

[56] T. Kudo and J. Richardson. *SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing*. 2018. arXiv: 1808.06226 [cs.CL].

[57] R. Řehůřek and P. Sojka. "Software Framework for Topic Modelling with Large Corpora". English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[58] Y. Li, J. Cai, and J. Wang. "A Text Document Clustering Method Based on Weighted BERT Model". In: *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. Vol. 1. 2020, pp. 1426–1430. DOI: 10.1109/ITNEC48623.2020.9085059.

[59] G. Lample and A. Conneau. *Cross-lingual Language Model Pretraining*. 2019. arXiv: 1901.07291 [cs.CL].

[60] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators*. 2020. arXiv: 2003.10555 [cs.CL].

[61] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].

[62] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020. arXiv: 1910.01108 [cs.CL].

[63] E. Loper and S. Bird. "NLTK: the Natural Language Toolkit". In: *CoRR* cs.CL/0205028 (July 2002). DOI: 10.3115/1118108.1118117.

[64] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Y. W. Teh and M. Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: JMLR Workshop and Conference Proceedings, May 2010, pp. 249–256.

[65] C.-Y. Fu, M. Shvets, and A. C. Berg. *RetinaMask: Learning to predict masks improves state-of-the-art single-shot detection for free*. 2019. arXiv: 1901.03353 [cs.CV].

[66] M. Röder, A. Both, and A. Hinneburg. "Exploring the Space of Topic Coherence Measures". In: WSDM '15. Shanghai, China: Association for Computing Machinery, 2015, pp. 399–408. ISBN: 9781450333177. DOI: 10.1145/2684822.2685324. URL: https://doi.org/10.1145/2684822.2685324.

[67] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. "Scikit-Learn: Machine Learning in Python". In: *J. Mach. Learn. Res.* 12.null (Nov. 2011), pp. 2825–2830. ISSN: 1532-4435.

[68] B. P. Knijnenburg, M. C. Willemsen, Z. Gantner, H. Soncu, and C. Newell. "Explaining the user experience of recommender systems". In: *User Modeling and User-Adapted Interaction* 22.4 (Oct. 2012), pp. 441–504. ISSN: 1573-1391. DOI: 10.1007/s11257-011-9118-4. URL: https://doi.org/10.1007/s11257-011-9118-4.

[69] M. Nilashi, D. Jannach, O. b. Ibrahim, M. D. Esfahani, and H. Ahmadi. "Recommendation Quality, Transparency, and Website Quality for Trust-Building in Recommendation Agents". In: *Electron. Commer. Rec. Appl.* 19.C (Sept. 2016), pp. 70–84. ISSN: 1567-4223. DOI: 10.1016/j.elerap.2016.09.003. URL: https://doi.org/10.1016/j.elerap.2016.09.003.

[70] P. Pu, L. Chen, and R. Hu. "A User-Centric Evaluation Framework for Recommender Systems". In: *Proceedings of the Fifth ACM Conference on Recommender Systems*. RecSys '11. Chicago, Illinois, USA: Association for Computing Machinery, 2011, pp. 157–164. ISBN: 9781450306836. DOI: 10.1145/2043932.2043962. URL: https://doi.org/10.1145/2043932.2043962.