Capstone Project

REPORT: Unsupervised Topic Modelling for Indeed

Mentor:

Julian Brooke

Authors:

Gurpreet Bedi

Simon Zheng

Lisa Liu

1. SUMMARY

Indeed is one of the most popular sites in the world, with users visiting each day, either to apply for a job or lookup for specific job-related articles. Since Indeed is an open-source website, it has both categories of users: registered and non-registered. Unsupervised topic modelling would help in enhancing the user experience and personalize the user's view, as it would automatically predict the reader attributes based on the user activity on Indeed. Due to the lack of supervised data with appropriate labels, we performed some Natural Language Processing (NLP) based analyses and then adapted an unsupervised approach to extract topics (i.e. Unsupervised Topic Modelling). This may further aid Indeed in using these attributes to recommend the user with relevant content.

Latent Dirichlet Allocation (LDA) is a widely used topic model and is often used in text classification. Gensim LDA contains several built-in functions that can help us generate meaningful numbers for comparison, such as topic coherence. We executed the parameter fine tuning and got the best topic model with the highest coherence. However, it is hard to decide the topic number only depends on the coherence because there could be topic overlapping. Hence we implemented both quantitative and qualitative evaluation approaches to evaluate the best topic number, which gives us 40 topics including some dominant ones such as medical, resume and interview, together with several new topics which do not appear in the smaller topic model, such as legal/ law, military, dress code, etc. These topics are representative and informative.

Although topic models such as Gensim based Latent Dirichlet Allocation (LDA), has shown to be a good starting point, there is a bit of effort involved through hyperparameter tuning to create meaningful topics. Therefore, we went ahead with the Neural approach to determine if we could generate stable and meaningful topics. We compared and evaluated transformer based pre-trained models such as LongformerModel and Sentence transformers model to determine which model provided semantically meaningful sentence embeddings which can further be used to generate stable topics. In order to achieve this we performed, compared and evaluated multiple dimensionality reduction algorithms such as PCA, t-SNE, UMAP and document clustering algorithms such as k-means, agglomerative clustering and HDBSCAN. Amongst these we finally chose UMAP and HDBSCAN for Neural topic modelling.

Based on an evaluation metric, we evaluated both Sentence transformers based topic model and Gensim LDA topic model. The results show that Sentence transformers based topic modelling outperformed Gensim LDA topic modelling in generating stable and meaningful topics and identifying semantically related articles.

2. DATASETS

The data provided by Indeed from the Indeed Career Guide site. All the data is in EN_US. There are five type of content (.json) files provided by Indeed, named as below:

- article.json
  - One of the Indeed pages, consisting of "free-form" articles managed by Indeed Editorial Team (Indeed, n.d., a). The majority of the content "types" fall into this category.

- There are a total of 13,639 distinct articles.
- Some of the example of category in articles include:
  - Career Development (Indeed, n.d., b)
  - Interviewing (Indeed, n.d., c)

- careerpathpage.json (Indeed, n.d., d)
  - Content under career highlights, the type of a career a person can pursue. All the content is organized and looks similar.
  - The content here has a total of 541 distinct career paths.
  - Some of the examples include:
    - Learn about being a Pharmacologist (Indeed, 2021a)
    - Learn about being a Forester (Indeed, 2021b)
- categorypage.json (Indeed, n.d., e)
  - One of the Landing pages, categorically highlighting different types of content eg: Resume Samples. These pages themselves don't have meaningful content nor are they intended to serve a particular audience.
    - There are a total of 24 distinct categories.
- coverletter.json and resumesamplepage.json (Indeed, n.d. f,g):
  - These are structured pages for sample cover letters and resumes, respectively.
  - There are 340 distinct CoverLetters and 344 distinct Resume Samples.
  - Some of the examples include:
    - Architect Cover Letter Sample
    - Research Assistant Resume Sample

Table1: JSON dataset field specifications

Field Name	JSON File Object Name	Description
ID	_id $oid	This corresponds to the respective document id.
Title	Possible object names: • title • contentTitle • h1	This displays the title of the given content type (***Note: Every JSON file has a title***)
Content	Possible object names: • content • primaryContent • contentA • contentA + contentB (concatenate the two fields)	This displays the content of the given title. (***Note: Every JSON file has a content***)
Locale	locale	Language used for this content

Category	category	Domain used to build the content route url
UrlRoute	urlRoute	Routes used to build the content route url

- Pageview matrix (Link Dataset):
  - It corresponds to the visitor count from one article to another article.
  - There are 190686 records with 3 columns and the details regarding the dataset fields have been provided in **Table 2.**

Table 2: CSV dataset field specifications

Field Name	CSV File Object Name	Description
Article URL	link_1	This corresponds to the Indeed article
Article URL	link_2	This corresponds to the Indeed article
Visit	visitor_count	This displays the content the visit count from link_1 to link_2

NOTE: The link dataset would enable us to perform semantic textual similarity between the articles, which can be evaluated using cosine similarity. This would in turn help Indeed recommend semantically related articles to the users in future.

3. METHODS

In order to assess the users' activity on Indeed Career Advice page and extract topics that may be of interest to the users, we adapted two main strategies, i.e. Gensim based Latent Dirichlet Allocation (LDA) and Neural Topic Modelling. Topic modelling is an unsupervised machine learning task. We try to find "abstract topics" that can describe document collections. This means that we have a collection of text, and we try to find patterns of words and phrases that can help us classify documents and group them by "topics".

3.1 LDA Topic Modelling

3.1.1 LDA

Latent Dirichlet Allocation (LDA) is a type of unsupervised machine Learning. Before we start, we don't know the subject of the document. We can only specify how many subjects to look for. However this is also an advantage relative to the neural approach, which could only produce 'soft' clusterings instead of the

'hard' ones provided by LDA. At the end, we can look at the results and find out if they are feasible or not. LDA points out that each document in the corpus is a combination of a fixed number of topics. A topic may produce a variety of words, including all the words observed in the corpus. These "hidden" themes are then surfaced based on the possibility of word co-occurrence. Formally, this is the problem of Bayesian reasoning.

Both Scikit-learn and Gensim have a very good implementation for the algorithm. The basic models of them are the same. The difference between them is the inference method. The inference method used by gensim is stochastic variational inference and the one used by scikit-learn is variational inference EM algorithm.

3.1.2 Scikit-learn LDA

To implement the LDA algorithm, we first transform our articles into vectors of numbers by applying the CountVectorizer class from the scikit-learn package (Scikit Learn, n.d.,a). Then we can apply the scikit-learn LDA model with several parameter patterns (Scikit Learn, n.d.,a). We also found the problem using it. First of all, it contains a number of overlapping topics. Secondly, if we want to dig deeper into these models, it is difficult to use scikit-learn LDA to generate some quantitative results for comparing among topics, such as topic coherence.

3.1.3 Gensim LDA

In addition to scikit learn, there are LDA models built in Gensim (Gensim Topic Modelling for Humans, 2021). The principles used are basically similar. We apply models.ldamodel to fine tune the parameters first and then get the best model (Gensim Topic Modelling for Humans, 2021).

Parameter tuning

Gensim library itself contains several built-in functions that can help us generate meaningful numbers for comparison, such as topic coherence. Topic coherence provides a convenient measure to judge how good a given topic model is. It measures a score that is ranged around 0 to 1, regarding a single topic by measuring the semantic similarity with top related words in the topic. It is helpful to distinguish between topics that are semantically interpretable topics and topics that are artifacts of statistical inference. There are several parameters which could be tuned to improve its value:

Table 3.1 gensim LDA parameter

Hyperparameter	value
alpha	0.01
eta	0.9
passes	50

- **num_topics**: this is the key of the model, which tells the best number of topics should be set.
- **alpha**: can be set to an 1D array of length equal to the number of expected topics that expresses a-priori belief for each topics' probability.
- **eta**: a-priori belief on word probability.
- **passes**: number of passes through the corpus during training.

We first set the num_topics equal to 10 and tune alpha, eta, passes. With these fine tuned parameters, we apply

Table 3.1.3.2 Coherence value of multiple topic numbers

topic number random_state = None	coherence value
10	0.641
20	0.655
30	0.634
40	0.636
50	0.642
60	0.647
70	0.599
80	0.606
90	0.605
100	0.585

LDA to our dataset to generate topics of the numbers we set. Then we can evaluate which is a good topic number to choose.

Then we set the parameter random_state = None, and check which model gives us the highest coherence value. After multiple iterations, we found that when the number of topics changes more than 10, the value of coherence has obvious differences. The results show that the model of topic 20 produces the highest coherence value of 0.655 among all the topic numbers from 10 to 100, which means that when topic number is 20, the topics are most distinct and representative. The coherence value of these models is close. However, it is still a big challenge for us to decide which number we should use as topic number, because it is hard to make sure every topic is exclusive and has no overlapping. It is lucky that Gensim has some built-in visualization functions that we can use to compare among different models.

3.2 Neural topic modelling

The steps taken into account to perform Neural topic modelling are as follows:

3.2.1 Neural model Embeddings

The first step in Neural Topic modelling was to convert the articles to numerical data i.e. embeddings. Transformer based models such as BERT which stands for Bidirectional Encoder Representations from Transformers have an attention mechanism that learns contextual relations between words in a text. It provides state-of-the-art embeddings. BERT models, especially pre-trained BERT models have previously shown to be successful in running various Natural Language Processing (NLP) tasks. Therefore, an algorithm involving BERT based semantic textual similarity sentence embeddings could help generate meaningful and stable topics. Longformerl and Sentence Transformers models were used to generate document embeddings for this analysis.

The intuition behind using LongformerModel is that transformer-based models such as BERT and RoBERTa have one major drawback, i.e., they cannot attend to long sequences. They have an input sequence length of 512 tokens. LongformerModel has a RoBERTa-base architecture and self-attention mechanism which scales linearly with sequence length and has the ability to process long sentences (Hugging Face, n.d.). LongformerModel has 4096 as an input sequence length.

The intuition behind using the Sentence Transformers model is that it can be used to derive semantically meaningful sentence embeddings which can be evaluated using cosine-similarity. As previously shown it reduces the effort of finding the most similar pair in just 5 seconds as compared to 65 hours with BERT/RoBERTa, while maintaining the same accuracy as BERT (Reimers & Gurevych, 2019)
Sentence Transformers model MS MARCO (version msmarco-MiniLM-L-12-v3 from huggingface) has a DistilRoBERTa-base architecture which provides a nice balance between speed and performance in comparison to other sentence transformers models (Hugging Face, n.d.). Sentence transformers has an input sequence length of 512 tokens.

Initially, off the shelf pre-trained LongformerModel and Sentence transformers Model were used to check how well the pre-trained models are performing on the current dataset (i.e. Indeed dataset). The intuition

behind using the pre-trained models is that pre-trained models are already trained on millions of tokens and have state-of-art embeddings, which could act as a benchmark in our case. In order to evaluate the embeddings generated by Sentence transformers models and LongformerModel, scikit-learn's cosine similarity was used as a metric on the link dataset provided by Indeed. The link dataset corresponded to the number of visits from one article to another article by the users. On the basis of cosine similarity scores, the best model was chosen for further analysis.

3.2.2 Dimensionality reduction and Document Clustering

In order to make sure that the articles that are semantically coherent are clustered together, such that, we could determine the topics within these clusters. We implemented various clustering algorithms, such as k-means, agglomerative and Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN). Since, clustering algorithms perform poorly with high dimensional data (curse of dimensionality), which work well at lower dimensions. Therefore, we also decided to try dimensionality algorithms such as Principal Content Analysis (PCA), t-distributed stochastic neighbor embedding (t-SNE) and Uniform ManifoldApproximation and Projection (UMAP) to reduce dimensionality of embeddings, so that the performance of the clustering could improve with low dimensionality.
Silhouette scores are generally used to measure the similarity of an object to its own cluster (cohesion) in comparison to other clusters (separation) (Scikit Learn, n.d.,b). The Silhouette scores for different dimensionality reduction algorithms were used as a measure for comparison and the algorithm with the highest silhouette score was then used to build the clusters (Scikit Learn, n.d.,b). Visualization of the different clusters were then used to identify the best model.

3.2.2.1 Dimensionality Reduction

Below are the different dimensionality reduction algorithms types analyzed:

❖ *Principal Content Analysis (PCA):*

PCA is one of the most commonly used unsupervised dimensionality reduction algorithms (Scikit Learn, n.d.,c). In order to reduce the dimensionality of high dimensional embeddings, it projects each data point onto the first few principal components to obtain lower dimensional embeddings while preserving as much of the data's variation as possible (Scikit Learn, n.d., c). It is a part of the Scikit-learn (sklearn.decomposition) library. Performed PCA by setting "n_components" value to 2, to determine if PCA could retain as much of the variation in the embeddings as possible, after the high dimensional embeddings have been reduced to 2 dimensional embeddings.

Hyperparameter	Value
n_components	2

Hyperparameter used for PCA is:

- **n_components:** It represents the number of components to keep, if it is not set then all components are selected.

❖ *t-Distributed Stochastic Neighbour Embedding (t-SNE):*

t-SNE is one of the statistical methods that takes high dimensional data and reduces it to low dimensional graph thereby retaining a lot of the original information. It does that by giving each datapoint a location in a two or three dimensional map. t-SNE is computationally very expensive. Therefore, in order to reduce the need for more processing power, first we had reduced dimensionality of the original embeddings using PCA (as a pre-processing step) to 50 principal components and then applied t-SNE. It is a part of the Scikit-learn (sklearn.manifold) library.

Hyperparameter	Value
n_components	50

Hyperparameter	Value
n_components	2

Hyperparameter used for t-SNE is:

- **n_components:** It represents the dimensions of the embedded space.

We set “n_components” of PCA equals to 50 and generate PCA reduced dimensionality embeddings (Scikit Learn, n.d., d). Then we would feed PCA reduced embeddings as an input for t-SNE to reduce the embeddings further by setting “n_components” equals to 2 (Scikit Learn, n.d., d). As mentioned in Scikit-learn documentation for t-SNE, for high dimensional data it is recommended to first decrease the dimensionality to a reasonable amount (e.g. 50) and then apply t-SNE as it uses more processing power (Scikit Learn, n.d., d).

Note: Since, document clustering algorithms suffer from the curse of dimensionality and perform poorly with high dimensional data to identify stable and meaningful clusters. Due to which we are performing dimensionality reduction. The intuition behind choosing n_components equals 2 for PCA and t-SNE is to visualize how well these dimensionality reduction algorithms can retain and represent the dataset at such a low dimension.

❖ *Uniform Manifold Approximation and Projection (UMAP):*

UMAP is a general-purpose manifold learning and nonlinear dimensionality reduction algorithm (McInnes, Healy & Melville, 2020). It is very effective for visualizing clusters or groups of data points and their proximities. The main difference between UMAP and t-SNE is scalability, it can be applied directly to sparse matrices, i.e. it eliminates the need to apply any dimensionality reduction such as PCA as a pre-processing step.Though it is similar to t-SNE but has higher processing speed and provides better visualization results.

UMAP has different hyperparameters which can have impact on resulting embeddings, the ones which we used for our dataset are:

- **n_neighbors:** It helps UMAP to maintain a balance between local vs. global structure, as with low values it forces UMAP to focus on local structure while higher values will make UMAP focus on larger neighbourhoods.
- **min_dist:** This helps UMAP to tightly pack the data points together, lower value means data points will be clustered closely and vise-versa.

Hyperparameter	Value
n_neighbors	15
n_components	10
min_dist	0.1
metric	cosine

- **n_components:** This allows the user to determine the dimensionality of the reduced dimension space.
- **metric:** This parameter controls how distance is computed in the ambient space of the input data.

We have experimented with different hyper-parameter settings to obtain the best out of the algorithm, and finally we were able to select some good values and the same has been specified in the table above. The dimensionality reduction model which performed better was chosen for further analysis. Results are provided in the Evaluation Section.

3.2.2.1 Clustering algorithms

Below are the different Clustering algorithms types analyzed:

❖ *K-Means:*

It is a method of vector quantization, that aims to partition n data points into k clusters, wherein each data point belongs to the cluster with the nearest mean (Scikit Learn, n.d., e). It is a part of the Scikit-learn(sklearn.cluster) library.

Hyperparameter	Value
n_clusters	6

Hyperparameter used for k-means is:

- **n_clusters:** It represents the number of clusters to form as the number of centroids to generate.

❖ *Agglomerative Clustering:*

Agglomerative Clustering is the most common type of hierarchical clustering used to group objects in clusters based on their similarity (Scikit Learn, n.d.,f). The main idea is that we start with each point in its own cluster and then, for each cluster, use some criterion to choose another cluster to merge with. We perform this activity repeatedly until we have only one cluster and we get a hierarchy, or binary tree, of clusters branching down to the last layer which has a leaf for each point in the dataset. It is a part of the Scikit-learn(sklearn.cluster) library.

Hyperparameter	Value
n_clusters	6

Hyperparameter used for agglomerative clustering is:

- **n_clusters:** It represents the number of clusters that the model should find.

Note: We used a sampled dataset for our evaluation. Since the default value of n_clusters for k-means is 8 and for agglomerative is 2 we decided that n_clusters equals to 6 would be a reasonable option to visualize the clusters generated by k-means and agglomerative clustering. Also, to visually depict if none of the clustering algorithms is forcing the datapoint to join either of the clusters.

❖ *Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN):*

Hyperparameter	Value
min_cluster_size	15
metric	euclidean

HDBSCAN is a fairly new dimensionality reduction algorithm developed by some of the same people who wrote the original DBSCAN (Density-Based Spatial Clustering of Applications with Noise) paper (HDBSCAN, n.d.). Their goal was to allow varying density clusters. We first transform the space according to density and perform single linkage clustering on the transformed space. We import the hdbscan library in order to use hdbscan. The clustering algorithm which performed better was chosen for further analysis.

Hyperparameter used for HDBSCAN algorithm are:

- **min_cluster_size:** It represents the minimum size of each cluster, and helps in automatically generating the clusters on the basis of its value (i.e. 15 here).
- **metric:** It is used to calculate distance between instances in a feature array.

3.2.3 Creating Topic words using cluster information

TF-IDF is used to determine the relevancy of words to a document in a collection of documents, which is achieved by multiplying two metrics: **term frequency** i.e. the number of times a word appears in the document and **Inverse document frequency** of the word across all the documents in the collection of documents (MonkeyLearn, 2019). Once the cluster has been generated, we need to know what makes one cluster different from the others. In order to achieve that we performed topic based TF-IDF, which would extract words associated with each topic. The intuition behind that is to compare the importance of words between the documents. The only difference with respect to standard TF-IDF is that instead of taking document by document, we would treat all the documents in one cluster as a single document and then apply TF-IDF. The result would be a very long document per topic and the resulting TF-IDF score would demonstrate the important words in a topic.

$$tfidf(t_j, T, D) = tf(t_j, T) * idf(t, D)$$

$$where, tf(t_j, T) = \frac{f_T(t_j)}{w_T} \; and \; idf(t, D) = log\frac{D}{\sum_i^n t_i}$$

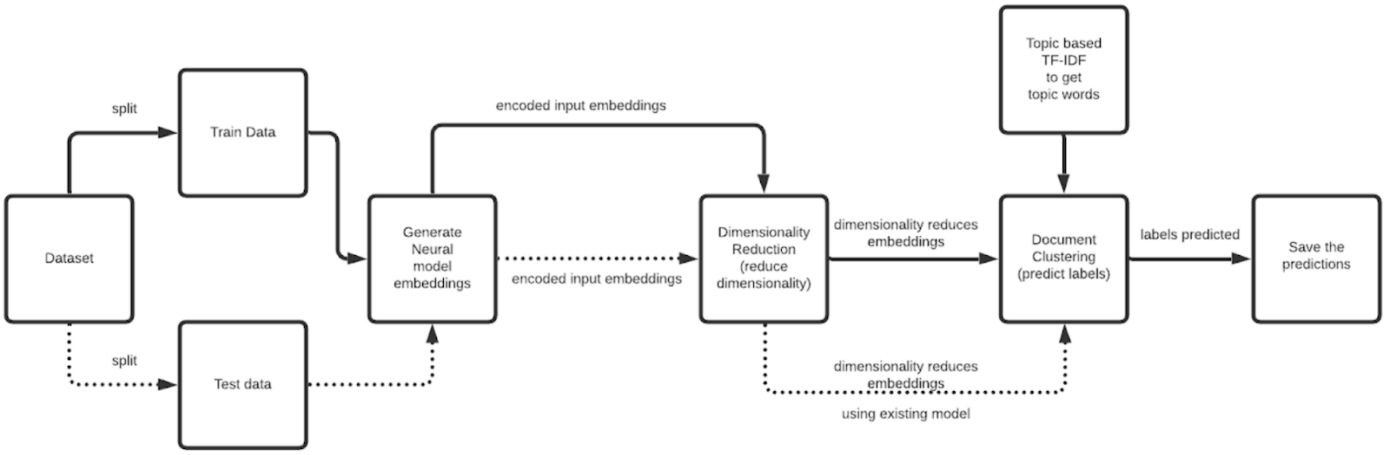
$f_T(t_j)$ = frequency of a term t_j in topic T

w_T = total number of words w in topic T

D = Total number of documents

The formula is that the frequency of each term “t” extracted for each topic “T” is divided by the total number of words “w” in that topic, this is required to normalize frequent words in the topic. This will form the term frequency (GitHub, n.d. a). After that, we need to calculate the inverse document frequency by taking the total number of documents “D” not clustered together, divided by the total frequency of each term “t” across all the topics, and finally we multiply them to get the TF-IDF score for each term.

Figure 1: FLOWCHART SUMMARIZING THE NEURAL TOPIC MODELLING PROCESS



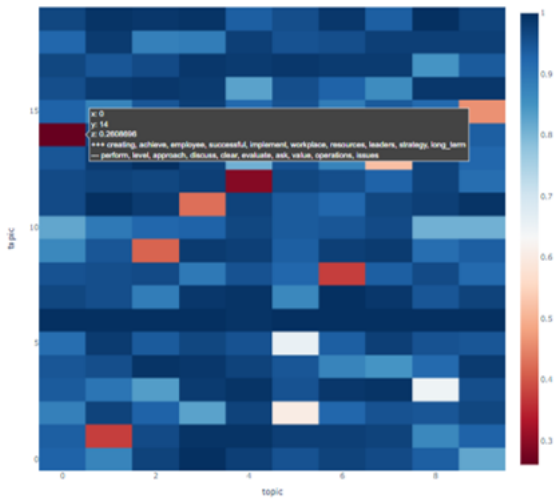
4. EVALUATION

4.1 Gensim LDA Topic Modelling

4.1.2 Heatmap(diff)

Gensim can help to visualize the differences between topics. To do this, we can use the diff() method of the LDA model. diff() returns the distance matrix mdiff and the annotated matrix annotation. mdiff [i] [j] refers to the distance between the topic i generated by the first model and the topic j generated by the second model.

Figure 4.1.1.1 Visualization of the topic distance between 10 and 20



First, we take a look at the heatmap of topic 10 and 20.
axis x: topics of model 10; axis y: topics of model 20
x= topic from model 10 ; y= topic from model 20
z= distance of x and y

The redder the grid, the smaller the difference between the themes, the bluer the grid, the larger the difference between the themes.

The basic idea of this comparison is that when we are comparing two models m and n (m < n), if the similarity between the two models is close to the value of m/n, and the extra topic of the n models are representative, we can tell that the model n is a good choice, unless that if we continue to compare model n with p, for one topic from model p, there are more than 1 similar topic from n to it, then we say that model n should be our choice. Then it is important to decide the threshold for the similarity between two topics from each model. We can eyeball the topic words of different z values to make the choice.

Table 4.1 value of topics from model 10 and 20

topic 10	topic 20	z value
(1, '0.022*"medical" + 0.022*"patients" + 0.012*"nursing" + 0.012*"health" + ' '0.012*"patient" + 0.009*"healthcare" + 0.008*"care" + 0.008*"nurses" + ' '0.006*"nurse" + 0.006*"primary_duties"')	(10, '0.021*"primary_duties" + 0.018*"clients" + 0.017*"manager" + ' '0.014*"managers" + 0.012*"financial" + 0.009*"professionals" + ' '0.009*"duties" + 0.007*"operations" + 0.007*"real_estate" + ' '0.006*"consultant"')	0.895
(4, '0.042*"resume" + 0.009*"relevant" + 0.009*"letter" + 0.008*"cover_letter" + ' '0.007*"qualifications" + 0.006*"applying" + 0.006*"candidate" + ' '0.006*"candidates" + 0.006*"email" + 0.005*"employer"')	(13, '0.021*"interview" + 0.015*"ask" + 0.007*"candidate" + 0.007*"candidates" + ' '0.005*"situation" + 0.005*"employer" + 0.005*"prepare" + 0.005*"tell" + ' '0.005*"employers" + 0.005*"asking"')	0.810
(5, '0.020*"marketing" + 0.020*"sales" + 0.019*"customers" + 0.015*"product" + ' '0.015*"customer" + 0.010*"products" + 0.008*"brand" + 0.008*"social_media" ' '+ 0.007*"clients" + 0.007*"content"')	(5, '0.042*"marketing" +0.022*"social_media" + 0.017*"content" +0.017*"brand" ' '+ 0.012*"event" +0.011*"website" + 0.008*"online" + 0.008*"advertising" + ' '0.007*"audience" + 0.006*"events"')	0.658
(7, '0.011*"interview" + 0.010*"ask" + 0.007*"meeting" + 0.005*"day" + ' '0.004*"tips" + 0.004*"situation" + 0.004*"personal" + 0.003*"asking" + ' '0.003*"workplace" + 0.003*"share"')	(13, '0.021*"interview" + 0.015*"ask" + 0.007*"candidate" + 0.007*"candidates" + ' '0.005*"situation" + 0.005*"employer" + 0.005*"prepare" + 0.005*"tell" + ' '0.005*"employers" + 0.005*"asking"')	0.519

After comparing the topic words of model 10 and 20, we can see that when the z value is 0.519, the topic words from each model are mostly alike, so that we set the threshold to 0.5. Then we can check the topic similarity between the two models as shown below.The similarity is the number of topics with z value smaller than 0.5 divided by n.From the results shown in the right table, we can see that the similarity between model 50 and 60 is the highest, however there are multiple matches from 50 for one topic from 60, for example for topic number 11, there are 10 similar topics from model 50, which is clear that the topics in model 50 are not distinct and representative.

Models	Similarity	Details
10/20	35.00%	
20/30	46.67%	
30/40	57.50%	
40/50	60.00%	
50/60 (0.602)	70.00%	0.0 (11, 5) 0.0 (11, 10) (截图(Alt + A) 0.0 (11, 25) 0.0 (11, 35) 0.0 (11, 40) 0.0 (11, 44) 0.0 (11, 45) 0.0 (11, 48) 0.0 (11, 49)

Table 4.2 Example of similar topics of model 50

topic number	topic words
5	(5, '0.000*"settle_disputes" + 0.000*"refuse" + 0.000*"paraphrasing" + ' '0.000*"overarching_goals" + 0.000*"josh" + 0.000*"interpersonal_conflict" + ' '0.000*"creating_implementing" + 0.000*"amicably" + ' '0.000*"willing_negotiate" + 0.000*"willing_accept"')
10	(10, '0.000*"settle_disputes" + 0.000*"refuse" + 0.000*"paraphrasing" + ' '0.000*"overarching_goals" + 0.000*"josh" + 0.000*"interpersonal_conflict" + ' '0.000*"creating_implementing" + 0.000*"amicably" + ' '0.000*"willing_negotiate" + 0.000*"willing_accept"')

11 …	(11, '0.000*"settle_disputes" + 0.000*"refuse" + 0.000*"paraphrasing" + ' '0.000*"overarching_goals" + 0.000*"josh" + 0.000*"interpersonal_conflict" + ' '0.000*"creating_implementing" + 0.000*"amicably" + ' '0.000*"willing_negotiate" + 0.000*"willing_accept"')

There are 10 topics from model 50 that match the topic 11 from model 60. The above table shows some examples of them, which are exactly the same. Hence, we can tell that model 40 is a better choice based on the heatmap results. Let's just compare the model 20(with the highest coherence value) with 40 to see if the larger one gives us more feasible topics. There are 15 similar topics between the two models, and the model 40 creates some possible topics which do not appear in model 20. There are some consistent topics which appear in each topic model, such as resume, medical/ nursing, construction/ equipment, marketing and interview related.

Table 4.3 Examples of similar topics between model 20 and 40

Model 20	Model 40
(12, '0.058*"resume" + 0.012*"relevant" + 0.011*"cover_letter" + ' '0.009*"qualifications" + 0.008*"applying" + 0.006*"title" + ' '0.006*"employer" + 0.006*"description" + 0.005*"employers" + ' '0.005*"accomplishments"')	(13, '0.062*"resume" + 0.012*"relevant" + 0.009*"qualifications" + ' '0.008*"applying" + 0.007*"title" + 0.006*"employer" + 0.006*"description" + ' '0.006*"accomplishments" + 0.006*"employers" + 0.006*"highlight"')
(1, '0.032*"medical" + 0.031*"patients" + 0.018*"nursing" + 0.017*"patient" + ' '0.015*"health" + 0.013*"healthcare" + 0.011*"nurses" + 0.011*"care" + ' '0.009*"nurse" + 0.006*"primary_duties"')	(21, '0.037*"medical" + 0.037*"patients" + 0.021*"nursing" + 0.020*"patient" + ' '0.015*"healthcare" + 0.014*"health" + 0.013*"nurses" + 0.012*"care" + ' '0.011*"nurse" + 0.007*"hospital"')
(3, '0.012*"construction" + 0.011*"equipment" + 0.011*"engineering" + "0.008*"primary_duties" + 0.008*"safety" + 0.007*"engineers" + '0.006*"systems" + 0.006*"technician" + 0.006*"design" + "0.005*"technicians"')	(25, '0.019*"construction" + 0.017*"engineering" + 0.015*"equipment" + "0.012*"engineers" + 0.011*"safety" + 0.009*"systems" + 0.009*"engineer" + ' '0.009*"technician" + 0.009*"design" + 0.008*"building"')
(5,'0.042*"marketing" + 0.022*"social_media" + 0.017*"content" + 0.017*"brand" ' '+ 0.012*"event" + 0.011*"website" + 0.008*"online" + 0.008*"advertising" + ' '0.007*"audience" + 0.006*"events"')	(26, '0.049*"marketing" + 0.025*"social_media" + 0.023*"content" + 0.019*"brand" ' '+ 0.013*"website" + 0.012*"event" + 0.010*"advertising" + 0.008*"media" + ' '0.008*"audience" + 0.007*"online"')
(13,'0.021*"interview" + 0.015*"ask" + 0.007*"candidate" + 0.007*"candidates" + "0.005*"situation" + 0.005*"employer" +0.005*"prepare" + 0.005*"tell" + "0.005*"employers" + 0.005*"asking"')	(28,'0.038*"interview" + 0.016*"ask" + 0.010*"candidate" + 0.008*"describe" + ' '0.007*"tell" + 0.007*"prepare" + 0.006*"situation" + 0.006*"interviewer" + ' '0.006*"employer" + 0.006*"employers"')

Other than the consistent topics, the model 40 also creates some representative ones which are not in topic 20, such as legal, human resource, military, dress code and cover letter related topics, which are really helpful. Although the coherence of model 20 is higher than that of model 40, the latter one creates extra more feasible topics, which is good for us to accept. Then we will check whether the model 40 could give us consistent results with multiple parameters, random_state, alpha and eta.

Model 40
(1, '0.023*"legal" + 0.019*"law" + 0.014*"case" + 0.012*"lawyers" + 0.011*"government" + 0.010*"lawyer" + 0.009*"law_enforcement" + 0.008*"law_school" + 0.008*"evidence" + 0.007*"cases"')

(6, '0.039*"employee" + 0.023*"hr" + 0.021*"policy" + 0.019*"human_resources" + ' '0.013*"leave" + 0.011*"policies" + 0.010*"employer" + 0.008*"supervisor" + ' '0.007*"workplace" + 0.007*"manager"')
(24, '0.029*"military" + 0.023*"officer" + 0.023*"army" + 0.015*"officers" + ' '0.013*"flight" + 0.012*"aircraft" + 0.010*"air_force" + 0.008*"navy" + ' '0.008*"pilot" + 0.006*"service"')
(36, '0.018*"wear" + 0.016*"hair_stylist" + 0.016*"hair" + 0.012*"jewelry" + 0.010*"skin" + 0.009*"clothing" + 0.008*"clients" + 0.008*"beauty" + 0.006*"makeup" + 0.006*"spa"')
(38, '0.127*"cover_letter" + 0.008*"cover_letters" + 0.002*"opening_paragraph" + 0.002*"closing_paragraph" + 0.002*"claims_adjuster" + 0.002*"time_consideration" + 0.001*"express_interest" + 0.001*"dear_hiring" ' '+ 0.001*"first_paragraph" + 0.001*"thank_consideration"')

Although the coherence of model 20 is higher than that of model 40, the latter one creates extra more feasible topics, which is good for us to accept. Then we will check whether the model 40 could give us consistent results with multiple parameters, random_state, alpha and eta.

4.1.3 Heatmap - multiple parameter

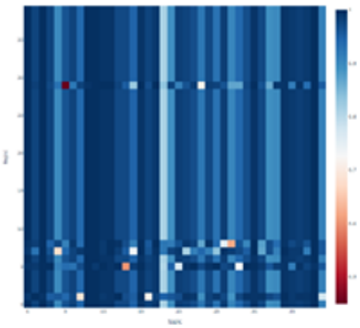
4.1.3.1 random_state

Previously we got the results from random_state = None. After comparing the coherence value with multiple random_state, we find out that different random_state values may produce various coherence numbers. Also, the heatmap tells us that the topics of those models are not consistent. When we set the threshold to 0.5, the similarity between two models is as shown in the right table:

Table 4.1.2.1 Similarity between model 40 with various random_state value

Model40 with different random_state	similarity
None/10	0.55
None/50	0.575
None/100	0.600

Figure 4.1.2.1 Model40 comparison with different alpha and eta



4.1.3.1 alpha and eta

When we set the alpha=10 and eta=10, there is only one similar topic between the two models, which is clear that new alpha and eta make the model inconsistent.

4.2 Neural Topic Modelling

4.2.1 Neural model Embeddings

In comparison to the Longformer Model, the Sentence transformer model produced embeddings that are of higher quality and typically work quite well for document-level embeddings. The Longformer model

provided high cosine similarity scores for all articles, irrespective of their semantic coherence. As seen in **Table 5**, the cosine similarity scores generated by the LongformerModel on the link data show high scores for the comparison between bad examples (i.e. the articles that are not semantically coherent).

Table 5: Cosine similarity scores for Longformer Model

Link1	Link2	Cosine Similarity Scores
7 Items To Bring to a Job Interview	What Is a Vet Tech?	0.996660
Q &A: Should you include a Cover Letter?	How To Develop a Training Plan for Your Team	0.996644
Semimonthly vs. Biweekly Pay Schedules: Understanding Key Differences	7 Life Lessons You Can Learn at Work	0.996608
7 Items To Bring to a Job Interview	Different Types of Benchmarking Examples	0.996590

The cosine similarity score for embeddings from the Sentence Transformer model were higher for articles that are semantically similar and lower for articles that were not semantically similar. Table 6, shows the results from cosine similarity score from Sentence Transformer model for the articles that were semantically coherent. The Sentence Transformer model also predicts higher cosine score when there is a higher number of visits from one article to another article by the users (See Table 7 and Table 8).

Table 6: Cosine similarity scores for Sentence Transformer model

Link1	Link2	Cosine Similarity Scores
Resignation Letter Due to Career Change: Tips and Examples	How To Write a Resignation Letter	0.911585
10 Resume Writing Tips To Help You Land a Job	How to Write a Resume Employers Will Notice	0.908975
Chronological Resume Tips and Examples	2021's Top Resume Formats: Tips and Examples of Three Common Resumes	0.893147
6 Universal Rules for Resume Writing (With Video)	10 Resume Writing Tips To Help You Land a Job	0.891810
The Essential Job Search Guide	Job Search Guide: Product Management and Software Engineering	0.888012
What does "Business Casual" Mean? (With Example Outfits)	What to Wear: The Best Job Interview Attire	0.880033

Table 7: Articles Ranked per visit data

Link1	Link2	Visit
List of Weaknesses: 10 Things To Say in an Interview	39 Strengths and Weakness in a Job Interview	2083
List of Weaknesses: 10 Things To Say in an Interview	How to answer “Tell me About Yourself” (Tips and Example Answers)	322
List of Weaknesses: 10 Things To Say in an Interview	Interview Question: “How Would You Describe Yourself?” (With Examples)	116
List of Weaknesses: 10 Things To Say in an Interview	Interview Question: “Why Should We Hire You?”	65
List of Weaknesses: 10 Things To Say in an Interview	Interview Question: “How Do You Handle Conflict in the Workplace?”	63

Table 8: Prediction on the link high ranked visits

Link1	Link2	Cosine Scores
39 Strengths and Weakness in a Job Interview	List of Weaknesses: 10 Things To Say in an Interview	0.754508
How to answer “Tell me About Yourself” (Tips and Example Answers)	List of Weaknesses: 10 Things To Say in an Interview	0.726102
How to Prepare for an Interview	List of Weaknesses: 10 Things To Say in an Interview	0.686610
14 Common Second Interview Questions (With Example Answers)	List of Weaknesses: 10 Things To Say in an Interview	0.669918
Problem-Solving Skills: Definition and Examples	List of Weaknesses: 10 Things To Say in an Interview	0.667006

We further evaluated different Sentence Transformers model such as:

- 'msmarco-MiniLM-L(12-v3 and 6-v3)
- 'paraphrase-MiniLM-L6-v2'

We ultimately chose **'msmarco-MiniLM-L-12-v3'**

4.2.2 Comparison of different Dimensionality Reduction Algorithms and Clustering Algorithms

Dimensionality Reduction Algorithms

Figure 2: Plot from Principal Content Analysis

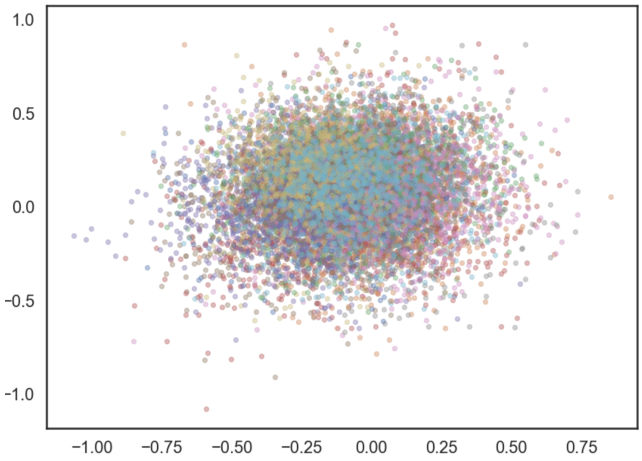
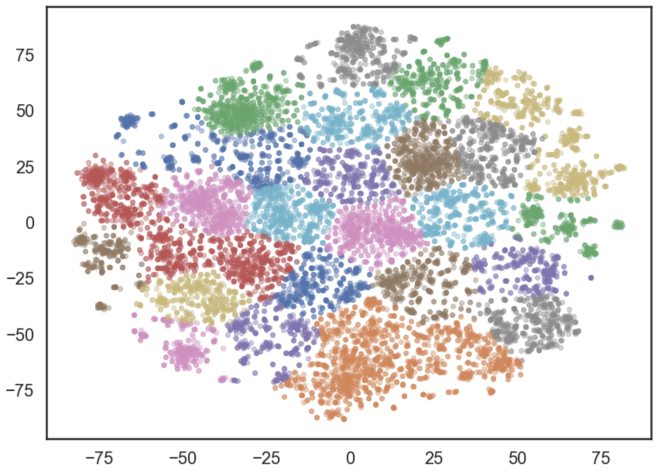
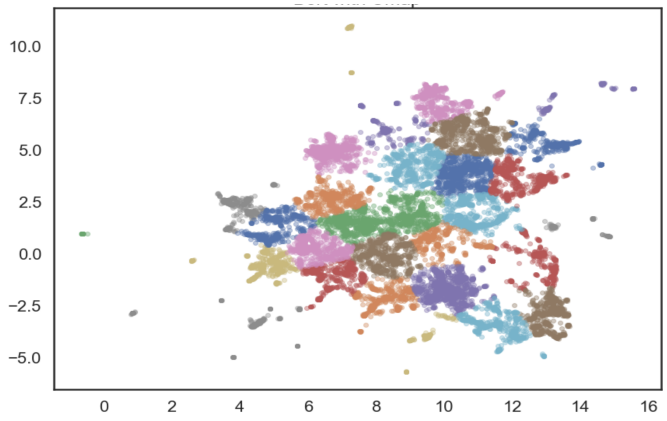


Figure 3: Plot from t-SNE



Among the dimensionality reduction algorithms implemented, UMAP is arguably the best performing algorithm as it has outperformed t-SNE and PCA. Figure 2,3, and 4 show the plots of the PCA, t-SNE and UMAP dimensionality reduction algorithms respectively. Plot from the UMAP algorithm shows some mini-clusters that are being separated well compared to other algorithms. It is very effective for visualizing clusters or groups of data points and their relative proximities. UMAP is much faster than t-SNE (See Table 10). Also, another problem with respect to t-SNE is the need for another dimensionality reduction as a pre-processing step, otherwise, it would take a longer time to compute (Scikit Learn, n.d., d).

Figure 4: Plot from UMAP



The colors in the figures 2, 3 and 4 represent the clustering performed on the top of the dimensionality reduced embeddings respectively. For testing purposes we used k-means as the clustering algorithm to generate non-arbitrary shaped clusters for better understanding of the dimensionality reduced embeddings.

Table 9 shows the Silhouette scores depicting the effect of different dimensionality reduction algorithms on Sentence Transformers embeddings. The Silhouette score for the UMAP was the highest compared to other algorithms. Thereby depicting it is effectively determining the similarity amongst data points within the cluster (cohesion) and in comparison to the clusters (separation) (Scikit Learn., n.d.,b). Silhouette score values range from -1 to 1 where 1 being the best and -1 being the worst. Therefore, UMAP was chosen to build the clusters (Scikit Learn., n.d.,b).

Table 9: Silhouette Scores for Sentence Transformers Embeddings	
Sentence Transformers Embeddings	**Silhouette Score**
No Reduction Applied	**0.04251874**
PCA	**0.32148919**
t-SNE	**0.3725367**
UMAP	**0.42552513**

Table 10: Computational Time for Dimensionality Reduction Algorithm.	
Dimensionality Reduction Algorithm	**Computation Time (seconds)**
PCA	**0.119**
t-SNE	**52.362**
UMAP	**5.147**

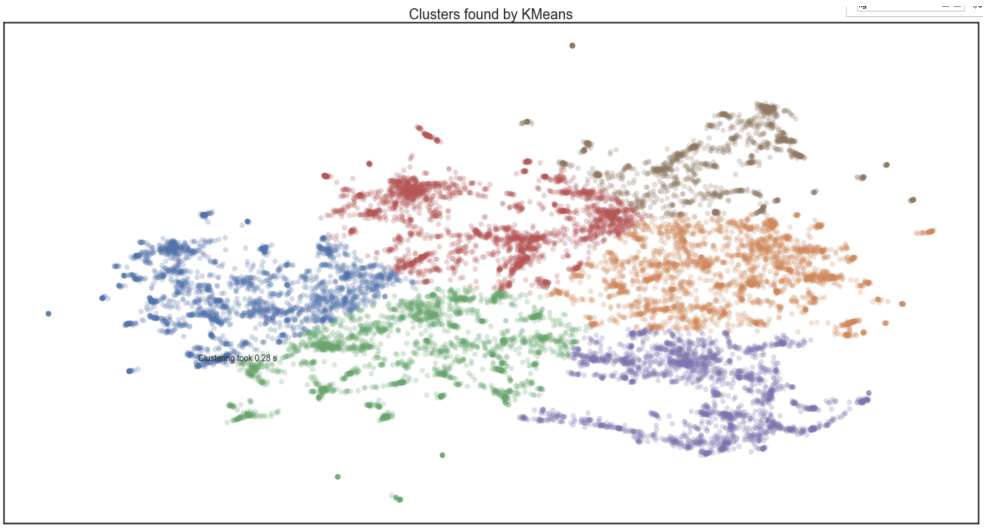
Note: The above computation results are generated using a Macbook machine running on CPU

Comparison of the Clustering Algorithms:

- *K-means:*

Figure 5 shows results from K-Means clustering. This figure depicts k-means clustering trying to join points to the cluster where the data points belong or not, which is not a requirement. Therefore, if we don't know how many clusters the dataset that we are exploring has, k-means would be a problem to use in our case.

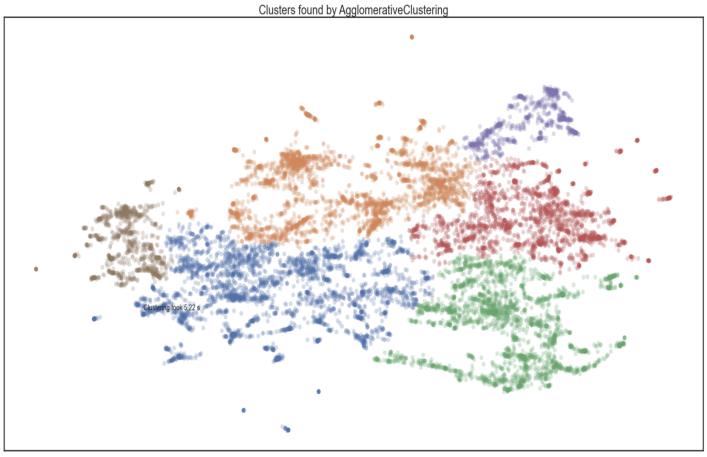
Figure 5: Plot from K-Means Clustering



❖ *Agglomerative Clustering:*

Figure 6 shows the representation of Agglomerative Clustering. It is similar to k-means, that is we have to choose a number of clusters (which is not easy in Exploratory Data Analysis), or try to recognize some parameter value from the plot that may or may not have any obvious natural choices.

Figure 6: Plot from Agglomerative Clustering



❖ *HDBSCAN:*

Figure 7 shows the representation of Agglomerative Clustering. It is clear that HDBSCAN, unlike k-means and Agglomerative clustering, is not forcing the data points to be clustered and the small black points on the graph represent the outliers. Therefore, if we don't know how many clusters the dataset that we are exploring has, HDBSCAN by default clusters the datapoints based on the different hyperparameters selected like minimum cluster size, distance metric, etc. HDBSCAN has inherited all the benefits of DBSCAN and is easily the strongest option as depicted in Figure 6.

Figure 7: Plot from HDBSCAN clustering

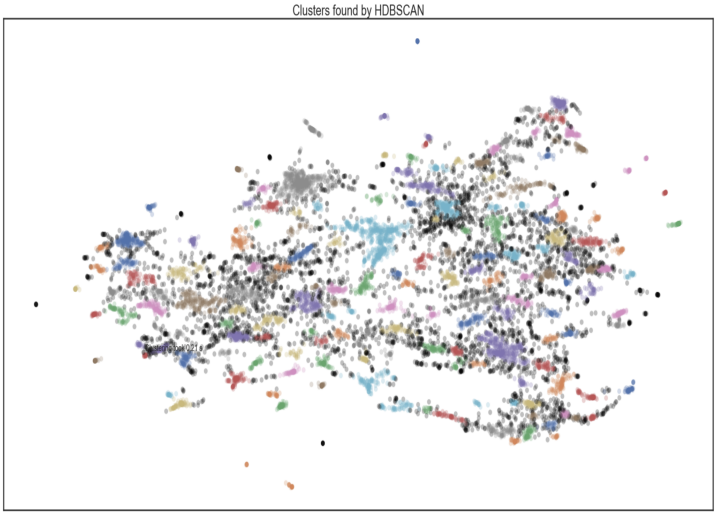


Table 11: Computation time for document clustering algorithm	
Document Clustering Algorithm	**Computation Time (seconds)**
k-means	**0.27**
Agglomerative Clustering	**5.46**
HDBSCAN	**0.20**

We tried performing Silhouette score as an evaluation and comparison measure to compare k-means, agglomerative clustering and HDBSCAN. The Silhouette score failed to evaluate HDBSCAN as they are

just capable of evaluating globular clusters and HDBSCAN form non-globular clusters (arbitrarily shaped). Therefore we used Density Based Clustering Validation (DBCV) algorithm which is used to evaluate the density of the data points within the clusters and density between the clusters.

Note: Executing DBCV is computationally expensive and takes time while calculating the density within the clusters and between clusters, as it performs a series of calculations to derive these results (GitHub, n.d. b). Therefore we used a subset of the dataset to derive the scores and HDBSCAN was selected for further evaluation (GitHub, n.d. b).

After performing UMAP for dimensionality reduction (reducing the articles dimensionality to **5),** we can then cluster the documents using HDBSCAN. HDBSCAN is a density based algorithm which works quite well with UMAP in comparison to other cluster algorithms like k-mean and agglomerative clustering, since UMAP maintains a lot of local structure even in lower-dimensional space. One of the advantages of HDBSCAN is that it does not force data points to clusters as it considers them outliers. Also, computation time for HDBSCAN is lowest in comparison to other clustering algorithms (See Table 11).

4.2.3 Creating Topic words

After performing category based TF-IDF, we create a topic representation with top 10 words per topic based on their category based TF-IDF scores. The higher the score, the more representative it should be of its topic as the score is the proxy of information density. One great thing about HDBSCAN is that not all documents are forced towards a certain cluster, if no cluster could be found, then it is simply an outlier.

The topic name -1 refers to all articles out of 13,639 that did not have any topics assigned (i.e. outliers). We can see that the topics 94, 120, 65, 140 are the largest clusters that we could create and also that we could see the words associated with those topics.

```
top_10_words[82]
```

```
[('interview questions', 0.002882094303765363),
 ('interviewer', 0.002713095369768925),
 ('answers', 0.0021949500516793654),
 ('question', 0.0021301921437067647),
 ('interviewers', 0.0018139703194763844),
 ('sample answers', 0.0016259538861313012),
 ('answer', 0.0015238723301266523),
 ('ask question', 0.0014983462730031657),
 ('describe', 0.0014594806727232278),
 ('sample', 0.0013983627807303168)]
```

Title	Topic
How to Introduce Yourself in an Interview	82
Data Scientist Interview Questions and Answers	82
Full Stack Developer Interview Questions (With Example Answers)	82
Interview Question: How Did You Hear About the Position?	82

```
top_10_words[84]
```

```
[('nurse', 0.009055076055919365),
 ('nurses', 0.008973646653876498),
 ('nursing', 0.008848611390205137),
 ('rn', 0.006094726271406565),
 ('registered', 0.005338051819788505),
 ('registered nurse', 0.004791529434091813),
 ('nurse practitioner', 0.004356038994123394),
 ('practitioner', 0.004150772426609827),
 ('bsn', 0.0038553064498498767),
 ('degree nursing', 0.0031909738199026715)]
```

Title	Topic
10 Highest Paid Nursing Jobs	84
52 Types of Nurses	84
How To Become a Radiology Nurse	84
What Is a Practical Nurse? LPN vs. CN vs. RN, Average Salary and How To Become a Practical Nurse	84

```
top_10_words[31]
```

```
[('resumes', 0.003945796576245943),
 ('summary', 0.0029251407006418376),
 ('chronological', 0.0025078425247189564),
 ('format', 0.0024939927042461 27),
 ('resume template', 0.002238068661660776),
 ('awards', 0.0021226876527861743),
 ('section', 0.002105445174638433),
 ('accomplishments', 0.0018946087554382385),
 ('template', 0.0018931670337302084),
 ('sections', 0.0018870715645685613)]
```

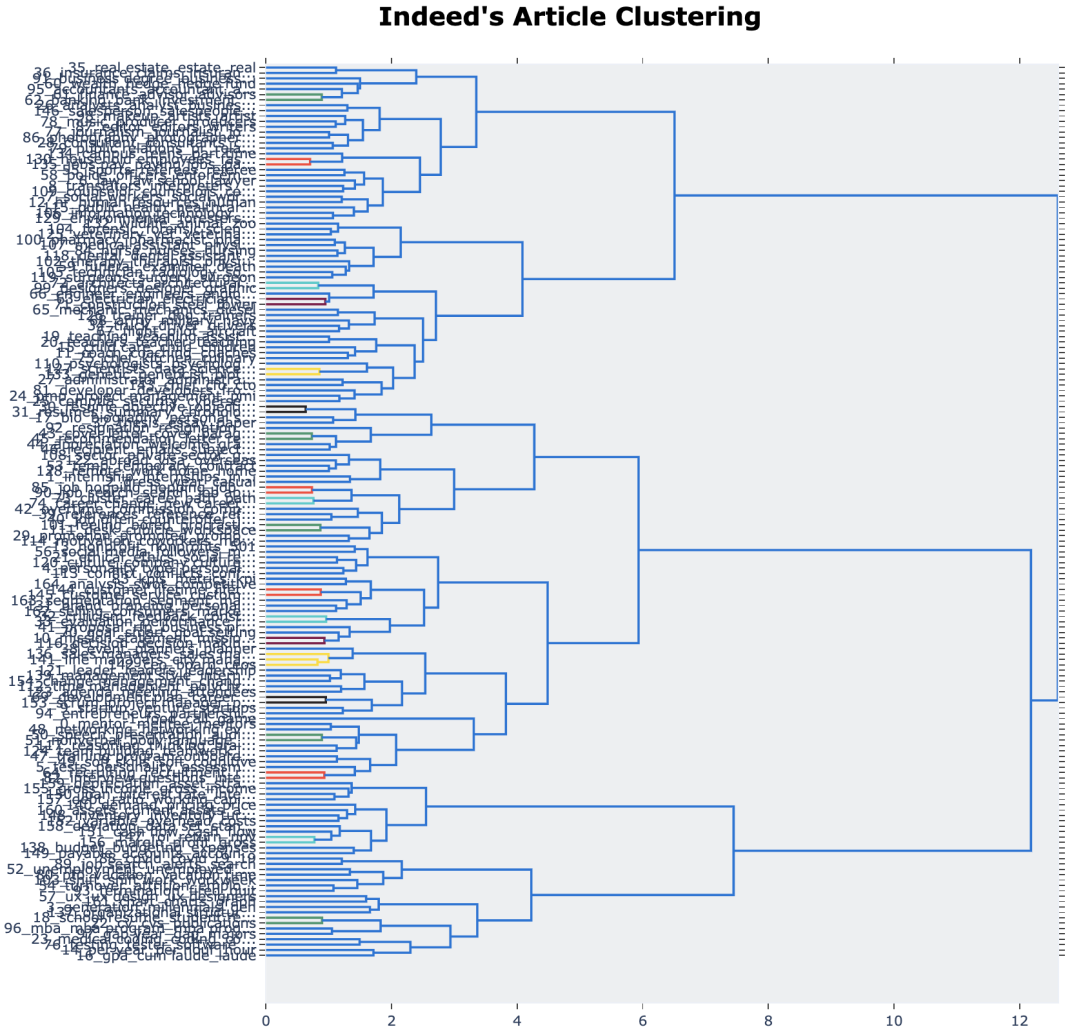
Title	Topic
How to Write a One Page Resume (With Example and Tips)	31
10 Common Resume Questions and Answers	31
10 Steps for Building a Resume	31
How To Prepare a Resume in 5 Steps	31

Looking at the largest four topics we could say that neural approach seems to nicely represent the interpretable topics i.e. **Interview Questions, Sales & Marketing, Resumes, Managers** as clear topics that were extracted.

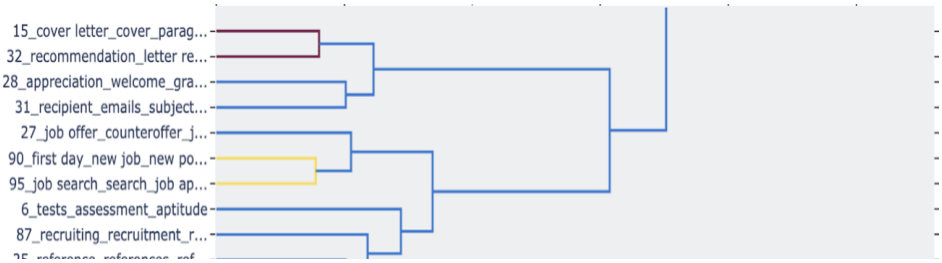
4.2.4 Dendrogram Visualization of Neural Topic Model

The dendrogram for a topic model clearly shows how well the articles which are semantically coherent are clustered together under one topic and all the topics that are related are placed under the same hierarchy. Figure 8 shows a dendrogram of all the Indeed article clustering in topic space.

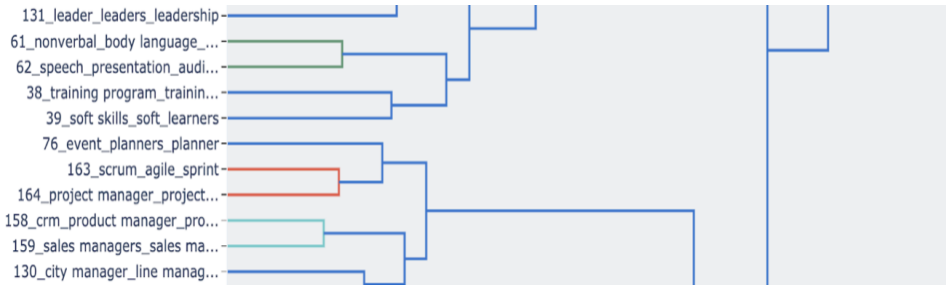
Figure 8: Full View of Dendrogram in topic space



For topics 15, 32 are the ones which show the notion of job related letters for eg: cover letter, recommendation letter



For topics like 158, 159 are the ones which show some notion of managers for eg: product manager, sales manager.



4.2.5 Evaluation metric

We used the article to article link data provided by Indeed to perform evaluations between Sentence transformers embeddings, TF-IDF embeddings and Gensim LDA Topic model representation.

Pageview matrix (Link dataset)

The Pageview matrix data have semantically coherent and related article pairs for higher number of visitor count, but there seems to be noise (random clicks) in the dataset for visitor count less than 5. Below is a result from the link dataset. Therefore, we would like to filter the dataset having visitor count >5, as it would be fair for the sentence transformer model as well and would not unnecessarily be biased and favour the other model.

link1	link2	visit
Digital Marketing Specialist	Supply Chain Manager	4
The Five Step Nursing Process	How To Write an Action Plan To Achieve Your Goals	4
5 Common Interview Questions About Conflict (With Example Answers)	8 Team-Building Activities for Improving Communication (With Tips)	4
How to Answer "What Motivates You?" (With Examples)	6 Examples of Critical Thinking Skills	4
12 Tough Interview Questions and Answers	4 Ways To Improve Organizational Climate	4
31 Tips for Working in a Call Center	Learn About Being an Accounting Assistant	4
Leadership Skills: Definitions and Examples	Functional Resume: Definition, Tips and Examples	4
Journeyman Electrician	Quality Assurance Analyst	4
10 Key Areas of Development for Employees (with Examples and Tips)	How to Write a Self-Appraisal	4
Medical Assistant Vs. Nurse: Differences and Similarities	9 Ways To Improve Your Personal Development Skills	4

Evaluation metric:

We performed the following steps in order to construct the evaluation metric:

1. We first started by creating a random shuffling and splitting the all the articles with an 80-20 split (80% being the train dataset and 20% being the test dataset).
2. After that we recorded the number of articles form train dataset which are present in "link_1" column of link dataset and number of articles form test dataset which are present in "link_1" column of link dataset.

3. We then calculated the cosine similarity of the embeddings for articles belonging to both the train and test dataset with every other article in the link dataset.
4. After which we generated the visit data matrix (which will hold visitor count) for both train and test articles with respect to all other articles in the link dataset. “1” was given to the indexes which were visited and “0” for the ones which were not visited.
5. Once the visit data matrix was constructed, we computed the ratio of average of visit to average of non-visit data for both train and test dataset. It was performed by first multiplying the cosine similarity scores with the visit link dataset, which would result in a sparse matrix and would contain just the cosine scores of the articles which were visited and then averaged that.
6. For the non-visit dataset, we inverted the visit link dataset to represent “0” for visit and “1” for non-visit and then multiplied the cosine similarity scores to get the sparse matrix representing the articles which were not visited and then averaged them.
7. Finally we divided the average of the visit data to non-visit data to get the ratio. This process was performed for both sentence transformer embeddings and TF-IDF embeddings.

Below are the results for the ratios of Sentence transformers and TF-IDF.

Sentence Transformers vs TF-IDF

Train Dataset:

Table 17.1: Results for ratio of visit to non-visit data	
Sentence Transformers	**2.395**
TF-IDF	**2.331**
Sentence Transformers (UMAP Reduced Version)	**2.747**
Gensim LDA	**1.214**

Test Dataset:

Table 17.2: Results for ratio of visit to non-visit data	
Sentence Transformers	**2.392**
TF-IDF	**2.348**
Sentence Transformers (UMAP Reduced Version)	**2.557**
Gensim LDA	**1.323**

Note: The dataset was filtered to have visit count greater than 5 to prevent the noise getting introduced in the dataset.

5. ANALYSIS

According to the results from Gensim LDA, it is hard to tell which is the exact number to choose for the model. It is important to combine quantitative and qualitative approaches to evaluate. Usually there is a trade off between the coherence value and the topic quality. Furthermore, comparing the neural models, LDA does not perform as effectively as it. Hence, with regard to difficulty of the implementation and the results, neural is a better model to go with for this project.

As per the results from the evaluation section, sentence transformers model embeddings are performing better in terms of capturing the semantic textual similarity between the documents in comparison to the TF-IDF and Gensim LDA Topic model. We also performed the evaluation of Sentence transformers model with UMAP reduced dimensionality and it seems that UMAP is very well maintaining the local structure of the dataset at such a low dimension and the results are comparable to Sentence transformers model with original embeddings and TF-IDF representations. It was expected as Sentence transformers model is built and trained with the expectation to perform semantic textual similarity search between the two documents whereas TF-IDF is a word based approach and determines how relevant a word is to a document in a collection of documents and calculates the score accordingly. Since there is not much difference between the results of Sentence transformers embeddings and TF-IDF embeddings which means that Indeed articles link data could have been derived using TF-IDF vector representation and the same is being used to recommend articles to the users.

We were surprised to see the results for Sentence transformers model embeddings with UMAP and HDBSCAN in case of neural topic modelling where they were used to determine stable and coherent clusters of documents (see in Figure 8). They were much better in representation in comparison to Gensim LDA topic modelling.

The problem statement by Indeed comprised of the following aspects:

- ***Capture reader attributes***: The neural topic modelling would help Indeed capture user attributes using the topics generated using sentence transformer embeddings.
- ***Recommend related articles to the user***: Sentence transformers embeddings would help search for semantically coherent and related articles using the information obtained from the user click.

We also created a Neural model application which could be used to perform training and make predictions. The script includes using Sentence transformer models embeddings to perform dimensionality reduction and document clustering. The script is interactive and offers options to the user for saving artifacts such as topic embeddings, interactive visualizations (as HTML) such as topics space visualization, topics dendrogram to understand the topics better.

There were some challenges involved with this analysis. As:

- For the link dataset, the number of article link pairs which were less than 5 visits contributed close to 92% of the dataset, most of which consisted of unrelated articles. It caused errors while evaluating the models i.e. sentence transformers model and TF-IDF during evaluation. We addressed the

problem by filtering the dataset having visits greater than 5 and construct an evaluation metric which could help determine the performance of neural model embeddings.

- There was no supervised data provided, we had to follow an unsupervised approach which created some challenges as we were not getting stable clusters using Gensim LDA topic modelling. We addressed the problem by trying different dimensionality reduction and document clustering algorithm and eventually we were able to find pair of dimensionality reduction and document clustering i.e. UMAP and HDBSCAN which were reducing the dimensionality by maintaining the local structure of the data and forming stable clusters.

6. FUTURE WORK

- Indeed could use the neural topic modelling approach to:
  - capture reader attributes.
  - recommend articles to the users.
- Fine-tune the pre-trained model for downstream tasks and re-evaluate the model.
- Use the cosine scores generated on link data to train the pre-trained model.
- Expand the existing sentence transformers model to accept 4096 input sequence length for encoding.
- Comparing and evaluating results of both expanded model and existing model.
- For Gensim LDA, include more articles to run with more topic numbers, see if LDA behaves close to Neural.
- Enhance the evaluation metric for testing different topic models

7. APPENDIX

TIMELINE

Below are the tasks listed for the project:

- ***Week 1 : Analyzing Datasets***
  - Analyzing the dataset
  - Reading the JSON files
  - Identifying the fields associated with each JSON file.
  - Perform data analysis
  - Manually annotate a small set of articles from article.json to identify user attributes and get this list of attributes approved from Indeed.
- ***Week 2 : Implementing Non-Neural Unsupervised Approach***
  - Create an algorithm for non-neural unsupervised approach to predict user attributes.
  - Non-Neural Topic modelling:
    - Scikit-learn LDA model
    - Gensim LDA model
    - Documenting the results
    - Comparing the results of non-neural unsupervised approaches.

        - Identifying the one which has a better quality.
- ***Week 3 + Week 4: Implementing Neural Unsupervised Approach***
    - Create an algorithm for neural unsupervised approach to predict user attributes:
    - Neural models comparison:
        - Pre-trained LongformerModel
        - Pre-trained Sentence Transformers
        - Documenting the results
        - Comparing the results of neural unsupervised approaches.
        - Identifying the one which has a better quality.
    - Create an algorithm for Neural Topic modelling:
        - Pre-trained Sentence transformers model based topic modelling
        - Documenting the results
        - Comparing the topic modelling results of non-neural unsupervised approaches.
        - Identifying the one which has a better quality.
- ***Week 5 : Hyper-parameterizing and Evaluation***
    - Hyper-parameterizing the machine learning model(s) for better results for both neural and non-neural models.
    - Evaluating the results from two different approaches (unsupervised and supervised) using Link data.
- ***Week 6 : Integration and Code Review***
    - Code Review:
        - Reviewing the teammates' code.
        - Code Cleanup (if required)
        - Integration
        - Integrating all the modules
        - End-to-end execution of the integrated code, to ensure successful completion of the project.
- ***Week 7 : Report writing and Presentation***
    - Developing a final report detailing all the project components.
    - Creating video presentations.
        - Video Flash Talk and Demo for General Public
        - Video Presentation for Partners

Weekly breakdown of the work assigned amongst team members:

- ***Week1***
    - ***Gurpreet:***
        - Worked on analyzing and understanding the dataset provided by Indeed, defining data preprocessing strategies.
        - Performing manually annotation and identifying the reader attributes from the dataset provided.
        - Explained the problem at hand to the team.
        - Developing scripts to identify the importance of words in each dataset.
        - Strategically identifying the parts of the data which could be used for performing non-neural and neural topic modelling.
        - Lead in creating teamwork contract

- Lead in creating a project plan.
  - ***Simon:***
    - Explored and analyzed the raw data from Indeed.
    - Implemented some pre-processing approaches. Like stripping html hyperlinks, and html tags.
    - Manual annotation on reader attributes. e.g intent & attributes slots.
    - Searching for examples that related to topic modelling. eg. LDA, neural
  - ***Lisa:***
    - Collaborate on team contract
    - Collaborate on the project plan
    - Data Exploration
- ***Week2***
  - ***Gurpreet:***
    - Created an efficient Gensim LDA topic modelling algorithm.
    - Optimizing Hyperparameters the model and determining topic coherence across multiple topics models i.e. from 10 to 100 and identifying the best one.
    - Guided Simon in creating Scikit-learn LDA topic modelling and comparison and evaluation with Gensim LDA topic model using topic coherence.
  - ***Simon:***
    - Started trying Scikit-learn LDA topic modelling.
    - Ran grid search for different hyperparameters for Scikit-learn LDA. Mainly focused on selecting the right topic numbers.
    - Run several different Gensim LDA models in Colab
    - Compared the Scikit-learn LDA and Gensim LDA
  - ***Lisa:***
    - Manual annotation and project plan update
    - Preprocessing update (remove html)
    - LDA pre-implementation
- ***Week3***
  - ***Gurpreet:***
    - Implementing a pre-trained LongformerModel algorithm.
    - Implemented multiple pre-trained Sentence transformers model algorithms.
    - Evaluating both the models using Link data by evaluating the model embeddings using cosine similarity.
    - Choose the best on the basis of evaluation of cosine similarity results from each model.
  - ***Simon:***
    - Gird search for different hyperparameters for Gensim LDA topic modelling.
    - Continued working on selecting for topic numbers for both Scikit-learn LDA and Gensim LDA
    - Explored the built-in functions from Gensim library to see if there are any functions that can help us analyze the topic number easier.
    - Manually compared the topic top-words for different number of topic numbers under Gensim LDA

- ***Lisa:***
  - Longformer pre-implementation
  - Try multiple approaches to find the best topic number for Gensim LDA
  - Manually compared the topics generated by the Gensim LDA model
- ***Week4***
  - ***Gurpreet:***
    - Created a Neural topic modelling using Sentence transformers model embeddings
    - Performed clustering the embeddings directly using different clustering algorithms.
    - Performed different dimensionality reduction algorithms to improve Silhouette scores.
    - Performed different document clustering algorithms to determine stable clusters and coherent and consistent topics.
    - Performed topic based tf-idf to understand topics words to determine topic coherence.
    - Guided Simon and Lisa to perform interactive visualization using mdiff in heatmap.
  - ***Simon:***
    - Interactive visualization for Gensim LDA. (different Heatmaps)
    - Checked the overlappings between different Gensim LDA models
    - Comparing the topic results with neural approaches with Gurpreet, this was when I started realizing neural actually worked better in our context.
  - ***Lisa:***
    - Gensim evaluation(heatmap.mdiff())
    - Gensim consistency evaluation(random_state, Doc2vec cdist)
- ***Week5***
  - ***Gurpreet:***
    - Guided Simon and Lisa to try multiple Gensim LDA topic modelling such as ldamodel and ldamulticore.
    - Optimizing Hyperparameters dimensionality reduction and document clustering algorithm, to determine stable clusters and proper predictions.
    - Created a Neural Topic model application which could be executed interactively using user inputs.
    - Created command line scripts for Neural Topic modeling application.
      - Created a separate python script for Neural model training and making predictions.
  - ***Simon:***
    - Evaluated Gensim LDA based on the cosine similarity of Doc2Vec representations.
    - Document pairwise cosine similarity comparison
    - Topic cosine similarity comparison based on the grouped documents
    - Utilize the link data with the cosine scores for each document, and see if there were any connections between cosine scores and visit number (answer is no!)
  - ***Lisa:***
    - Gensim consistency evaluation (alpha and eta)
    - Gensim multiple iterations to double check the results
- ***Week6***
  - ***Gurpreet:***

- Performed code review of the Neural and Non-neural approaches and created separate python scripts for importing libraries and data pre-processing
- Guided Simon through the notebooks to clean the code by using methods and classes from Python files, instead of using repeating code over and over again.

○ ***Simon:***

- Changed Doc2Vec representations to topic distribution representations for documents. This embedding contains all the articles with their distributions over every topic. And then do the same thing like week 5.
- Assisted Gurpreet with the Neural topic model command line application by generating Python files that contain the workflow of the Neural topic modelling.
- With Gurpreet, Went through the notebooks we have so far, try to clean the code by using the methods and classes from Python files, instead of using repeating code over and over again.

○ ***Lisa:***

- Draft final report for Gensim LDA topic modelling
- Collaborate on code product wrap-up

- ***Week7***
  - ***Gurpreet:***
    - Created the Report and all the sections with respect to Neural Topic model and Evaluation
    - Created an evaluation metric which could be used by other team members for LDA topic modelling .
    - Formatted the entire report.
  - ***Simon:***
    - Created and filled out the Slides with Lisa for presentation.
    - Filling out the reports for Gensim part with Lisa, as well as the evaluation part for Gensim LDA.
    - Corrected and re-implemented the evaluation part for Neural with Gurpreet, changed the compute algorithms and delivered reasonable ratio scores for zero and non-zero visit number.
  - ***Lisa:***
    - Finalize the report for Gensim LDA topic modelling
    - Flash talk slides and presentation recording
    - Presentation for partner slides and recording
    - Collaborate on final code product wrap-up

REFERENCES

Gensim Topic Modelling for Humans. (April 29, 2021). *Latent Dirichlet Allocation*. https://radimrehurek.com/gensim/models/ldamodel.html

McInnes L., Healy, J., Melville, J. (2020). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *Stat ML.* doi: arXiv:18-2.03426.

HDBSCAN. (n.d.). *The hdbscan Clustering Library.* https://hdbscan.readthedocs.io/en/latest/index.html

Hugging Face. (n.d.). *Longformer*. https://huggingface.co/transformers/model_doc/longformer.html

Hugging Face. (n.d.). *Sentence Embedding Model for MS MARCO Passage Retrieval*. https://huggingface.co/sentence-transformers/msmarco-MiniLM-L-12-v3

Indeed. (n.d., a). *Browse Articles.* https://www.indeed.com/career-advice/browse-articles

Indeed. (n.d., b). *Career Development.* https://www.indeed.com/career-advice/career-development

Indeed. (n.d., c). *Interviewing.* https://www.indeed.com/career-advice/interviewing

Indeed. (n.d., d). *Explore Potential Career Paths.* https://www.indeed.com/career-advice/careers

Indeed. (2021a). *Learn About Being a Pharmacologist.* https://www.indeed.com/career-advice/finding-a-job/pharmacologist

Indeed. (2021b). *Learn About Being a Forester.* https://www.indeed.com/career-advice/finding-a-job/forester-job

GitHub. (n.d. a). *cTFIDF.* https://github.com/MaartenGr/cTFIDF

Indeed. (n.d., e). *Resume samples and templates to inspire your next application.* https://www.indeed.com/career-advice/resume-samples

Indeed. (n.d., f). *Cover Letter Samples and Templates to inspire your next application.* https://www.indeed.com/career-advice/cover-letter-samples

Indeed. (n.d., g). *Resume samples and templates to inspire your next application.* https://www.indeed.com/career-advice/resume-samples

MonkeyLearn. (2019). *What Is TF-IDF?* https://monkeylearn.com/blog/what-is-tf-idf/

Reimers, N., Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *Computation and Language.* doi: arXiv:1908.10084v1

Scikit Learn. (n.d.,a). sklearn.decomposition.*LatentDirichletAllocation*. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html

Scikit Learn. (n.d.,b). *Sklearn.metrics.silhouette_score.* https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

Scikit Learn. (n.d., c). *sklearn.decomposition.PCA*. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html?highlight=pca#sklearn.decomposition.PCA

Scikit Learn. (n.d., d). *Sklearn.manifold.* https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html?highlight=tsne#sklearn.manifold.TSNE

Scikit Learn. (n.d., e). *Sklearn.cluster.k_means.* https://scikit-learn.org/stable/modules/generated/sklearn.cluster.k_means.html?highlight=k%20means#sklearn.cluster.k_means

Scikit Learn. (n.d.,f). *sklearn.cluster.AgglomerativeClustering.* https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html?highlight=agglomera#sklearn.cluster.AgglomerativeClustering

GitHub. (n.d. b). DBCV. https://github.com/christopherjenness/DBCV