

**DASC7600 Data Science Project**

**COVID-19 Literature Analysis Based on NLP**

by

**Jinhui HU**

胡金辉

(University Number: 3035678105)

**Haozhe XU**

徐浩哲

(University Number: 3035678193)

**Ziqi Zhou**

周子奇

(University Number: 3035676858)

Supervisor: Professor Guosheng Yin & Dr. Ruibang Luo

Abstract of Final Report entitled  
**“COVID-19 Literature Analysis Based on NLP”**

Submitted by

**Jinhui HU**

(University Number: 3035678105)

**Haozhe XU**

(University Number: 3035678193)

**Ziqi Zhou**

(University Number: 3035676858)

In light of the recent COVID-19 outbreak, the coronavirus related literature rapidly increased. This study tries to use NLP tools to help health professionals quickly find related papers. We successfully trained a BERT model which has better performance on coronavirus related articles. We test this model on different datasets, and it gets a great result. The combination of BERT and K-Means realizes the extraction of text features, vectorization representation, and clustering. The approach divides a large collection of articles into several categories according to their abstract semantics and labels them. Based on the abstract's semantic features, cosine similarity realizes the similarity matching function by using the feature vector of the abstract. LDA obtains the probability of documents belonging to each cluster as the vector of sentence level. At the same time, BERT gets the whole abstract's embedding as the vector of the token level. After the combination of the two, K-means clustering is carried out to achieve a better clustering effect. This study has built a website that users can upload files and quickly find the related papers and their clusters. And also, users can easily view the results of our models on the website.

## List of Figures and Tables

### Figures

<b>Figure 1.</b> The number of confirmed COVID-19 cases.....	5
<b>Figure 2.</b> Part of raw data .....	12
<b>Figure 3.</b> Raw Data After Selection .....	14
<b>Figure 4.</b> The process of data preprocessing .....	15
<b>Figure 5</b> Frame of AutoEncoder.....	18
<b>Figure 6</b> Words in different Toics .....	21
<b>Figure 7</b> A generated article .....	22
<b>Figure 8</b> The structure of LDA.....	23
<b>Figure 9</b> Diagram of Unigram model .....	26
<b>Figure 10</b> Flow of Unigram model.....	26
<b>Figure 11</b> Diagram of Mixture of unigram model.....	27
<b>Figure 12</b> diagram of pLSA.....	30
<b>Figure 13</b> the Example of EM Algorithm .....	34
<b>Figure 15.</b> The visualization of LDA+BERT+K-Means.....	42
<b>Figure 14.</b> K-Means.....	50
<b>Figure 15.</b> DBSCAN .....	51
<b>Figure 16.</b> BIRCH .....	51
<b>Figure 17.</b> BERT .....	55
<b>Figure 18.</b> SCIBERT .....	56
<b>Figure 19.</b> SCIBERT-C19-1.0.....	56
<b>Figure 20.</b> SCIBERT-C19-2.0.....	56
<b>Figure 21.</b> Text matching process .....	59
<b>Figure 22.</b> The First Method's Flow Chart of Our Website.....	63
<b>Figure 23.</b> The Second Method's Flow Chart of Our Website .....	64
<b>Figure 24.</b> The screenshot of our website.....	66

### Tables

<b>Table 1.</b> The coherence of two model .....	42
<b>Table 2.</b> Table of Clustering Performance.....	54

# **1. Introduction**

## **1.1 Project background**

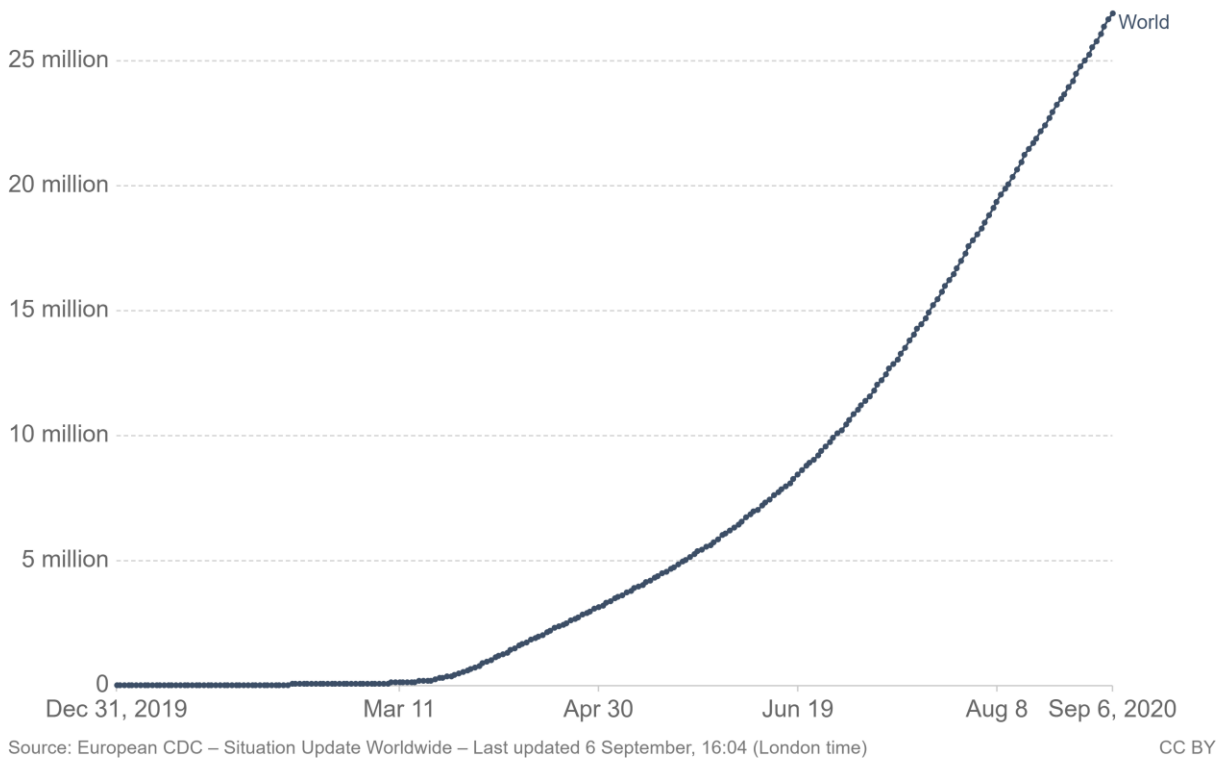
At the beginning of 2020, a very contagious virus broke out worldwide, called Severe acute respiratory syndrome coronavirus 2, or SARS-CoV-2 for short. In mid-January 2020, China discovered the virus through nucleic acid testing and genome sequencing of a positive patient sample and reported it to the World Health Organization. As of November 14, 2020, 191 countries and regions worldwide have reported more than 53.021 million confirmed cases, of which more than 1.29 million have died, and 34.26 million have been cured. The number is still rising rapidly. Currently, there is no effective vaccine or antiviral drug for COVID-19. The World Health Organization (WHO) declared the COVID-19 outbreak as a public health emergency of international concern (PHEIC) on January 30, 2020, and assessed on March 11, 2020, that the COVID-19 disease has a pandemic feature. The rapid spread of the virus has plunged the world into a dangerous situation. Borders have been blocked, and the flow of people has been restricted. Especially in the current globalization, the world's economy has fallen into a great dilemma. According to the World Bank's forecast, the new crown epidemic will exacerbate the increase in the world's extremely poor population, and global GDP will fall by 5.2%, the largest drop since World War II. The epidemic and the economic crisis have become the new root cause of the world's problems, which has plunged the world into crisis.

Researchers worldwide have taken action together to use their expertise to help all humanity through this crisis. Many scientific researchers study vaccines, infection mechanisms, data analysis, etc., and produce thousands of papers (including English papers only) every day. The rapid information explosion prevents front-line researchers from quickly grasping the latest in the fields they need. The research progress puts everyone in the information unequal stage, resulting in different research levels, and time was greatly wasted. With the rapid development of the epidemic, and sometimes the speed of virus updates is faster than the speed of information dissemination, everyone cannot better relay and contribute to the fight against the epidemic. Therefore, an urgent problem is in front of everyone, which is a good system and algorithm for document clustering and document recommendation are needed to help researchers save time, quickly grasp the latest research progress, and then be able to carry out more quickly their own scientific researches and contribute to fighting the epidemic and saving the common destiny of humanity.

With the rapid development of natural language processing technology, many systems for clustering and searching by analyzing article titles have appeared on the market, which has provided great help for researchers to find effective information. But now, in the face of the information explosion, the traditional engine that only analyzes article titles can no longer meet everyone's needs for information retrieval. The lack of optimization for specific fields has caused statistics and recommendation functions for current novel coronavirus-related literature is limited, unable to analyze the information in the article perfectly. Simultaneously, since the system only analyzes the document's title, it is easy to cause deviation. For these reasons, even if there is a literature search system, this system cannot serve everyone well. Therefore, how to optimize the current system and provide better help for researchers has become a major problem. Our project is to solve this dilemma, compete for time for front-line researchers, and improve efficiency.

### Cumulative confirmed COVID-19 cases

The number of confirmed cases is lower than the number of actual cases; the main reason for that is limited testing.



**Figure 1.** The number of confirmed COVID-19 cases.

## **1.2 Motivation**

Based on the above motivations, this research is committed to creating a model dedicated to natural language processing for the new coronavirus and building a friendly and easy-to-use document recommendation engine for use by researchers worldwide. The objectives of the project are as follows:

1. Train a Bert model with a good recognition rate for the novel coronavirus-related literature so that it has a better effect on the novel coronavirus than the basic Bert model so that users can get more accurate results;
2. Combine the Bert model and the LDA model to cluster our literature data, and extract the main meaning of the cluster by extracting keywords in the cluster, so that users can better understand the meanings of the clusters, and it is also convenient for users to locate their interests quickly;
3. Use mature and stable network technology to build a light and easy-to-use website, which can quickly and conveniently provide users with similar articles and related clustering information, including keywords, etc., and also the similar articles need to be located quickly, users can really easily get the information they need from the website.

Through these several purposes, we can roughly solve the dilemma that we mentioned in the previous part due to the outbreak of the epidemic, and truly provide scientific researchers with a convenient, fast, accurate and reliable document clustering and recommendation system, which can improve the efficiency of front line scientific researchers' search for relevant documents which can, in turn, help scientific researchers quickly familiarize themselves with the current research dynamics, and in a certain sense, solve the current dilemma of the information outbreak caused by the new coronavirus. At the moment, efficiency means life, and high efficiency can help the world.

## **1.3. Problem identification**

The project mainly involves the following issues.

1. Data collection problem. With the outbreak of the new coronavirus, related documents have been published in large numbers. However, many documents have different copyright requirements, which are difficult to collect and analyze. It takes a lot of time

to collect these documents, and the amount of data is huge and growing rapidly. It is easy to cause the collection speed to be far inferior to the publication speed and cause the database to become outdated. On the contrary, our system cannot provide users with timely and accurate information. Therefore, how to quickly collect data and update the data promptly is a big problem for us;

2. The problem of model building. Nowadays, the relevant natural language processing technology is relatively advanced. Still, there is no model related to the new coronavirus, so building a natural language processing model independently will also be a big problem. How to train an accurate model and evaluate it will be the most important part of the entire project, and our entire system needs not only a model, how to coordinate the corresponding results of multiple models will also be one of the problems we face;
3. The construction of the website. Although the relevant technologies and frameworks for website building are relatively mature now, how to build a friendly and easy-to-use website will still be a problem for us. One of the problems is how to reduce the cost of operating the website as much as possible so that our entire project is available to everyone for free to help the vast number of scientific researchers better. The second is how to build our entire front-end, back-end, and database to expand better and increase to cope with the website's possible development in the future. These two problems will inevitably be encountered in the process of building our website system.

#### **1.4. Significance**

In this project, we developed a complete document recommendation system and trained a natural language processing model that can target the new coronavirus. In general, the results we have obtained have great academic and practical significance, as follows.

The project's academic significance is to train a natural language processing model that can target the new coronavirus. Compared with the previous basic Bert model and the Sci-Bert model specially created for scientific research texts, our model is more effective against the new coronavirus. With a good improvement and a high degree of adaptation, we can quickly call our model for further analysis and processing of related documents.

This project's practical significance is to build a friendly, convenient, and free document search and clustering recommendation system. Researchers can quickly search for keywords and related literature through this system. It is easy to locate the related literature we provide, which greatly improves scientific research efficiency on literature search and can help solve the information explosion situation caused by the epidemic's development.

### **1.5. Literature review**

In the decades of the development of natural language processing, along with wave after wave of new technology, researchers have proposed various text feature extraction methods, but there has always been a big gap between the effect of manual extraction. With the outstanding performance of deep learning in the field of artificial intelligence, text feature extraction continues to achieve good results. The following is an introduction to the current research status of text feature extraction at home and abroad from two perspectives of unsupervised and supervised.

Unsupervised methods are mainly based on fine-grained methods, including statistical feature extraction methods, methods of constructing weighted graphs, and methods based on topic models. Because the unsupervised method does not require human involvement, it is more suitable for processing large data sets and extracting features from articles, both in terms of speed and efficiency, with higher automation effects. But there are also shortcomings such as manual adjustment of model parameters and the stitching of the later extracted text. Because it is only a simple extraction method, there is still a gap in the effect of comparing manual abstract extraction.

Based on the supervised method, the main idea is to use the manually distinguished labeled document set as the training set, using machine learning methods or deep learning methods for training, through training a complete end-to-end model, so that the machine can generate Artificial text feature recognition function.

Barzilay proposed the Lexical Chain method to find multiple important Lexical Chains by introducing the synonym table in semantic dictionaries such as WordNet to extract characteristic sentences in the text.



Word vector is an essential component in natural language processing. The existence of word vectors can be seen in almost every NLP task. The word vector uses the articles in the corpus to predict the current text by constructing an N-Gram language model combined with the context, or using the current text to predict the context, and finally based on the weight matrix obtained by the language model training and the word and One-Hot point Product, get the word vector corresponding to each word.

As the input of deep learning methods, word vectors have continuously improved their ability to represent semantics, which has also promoted the development of deep learning, allowing researchers to continuously update various deep learning methods and frameworks. Deep learning is more because of other methods. The key is that it can transform text into a continuous real-valued fixed dimensional vector by learning the types of artificial recognition, so as to avoid the use of external resources or artificial experience. Setting the weight causes the dependence of tuning. There are currently two types of deep learning in the field of text feature extraction.

One is to use the powerful structure of deep learning for extractive text feature extraction. Yan converts document segmentation into word vectors for input, and hidden layers uses multiple RBMS to build deep structures. However, this type of deep learning framework generally builds an extractive query framework based on semantic similarity.

Another type of deep learning framework generates features by training Seq2Seq. Seq2Seq was first proposed by Sutskever. Encoding is performed by using the Seq2Seq idea of input and output, and the encoding process is a process of supervised learning, and finally the entire article is predicted through the model to output features.

Facebook's Rush used the CNN model for the first time to encode and decode the original text. Taking into account the lack of information in the full text of the article, it introduced an attention mechanism and achieved good results on the DUC dataset.

Then Chopra introduced RNN for encoding by comparing with CNN, hoping to obtain more sequence context information, and the effect was greatly improved.

In 2016, Google open sourced the automatic title module Textsum in the deep learning framework Tensorflow, and improved the effect of automatic summary generation by introducing Beam Search and using the idea of the Seq2Seq framework. In 2017, Facebook and Google published 2 papers based on Seq2Seq. Among them, Jonas Gehring of Facebook proposed to use pure CNN to replace RNN for Seq2Seq, while Ashish Vaswani of Google proposed the concept of Attention, and proposed a variety of Attention mechanisms.

Salon proposed the TF-IDF method. The basic idea is that for articles in the corpus, the frequency of the word appearing in the article is positively correlated with the importance of the word, and negatively correlated with the number of documents containing it. By introducing the importance of TF-IDF word frequency, it is possible to better identify which are the important words in the article and which are the commonly used stop words, which improves the effect of word frequency extraction.

At present, there are many researches on the extension of LDA model, including vocabulary based extension model, time evolution based extension model, hierarchical relationship based extension model, sentiment analysis based extension model, short text based extension model, tag based extension model and comparative text mining based extension model. I mainly read some paper about vocabulary based extension model.

There are many theories supporting LDA model, among which bag of words model is a very important one. In this theory, a document is regarded as a collection of words. It ignores the order of words and syntax of sentences. It can only use some unordered words to express the topic extracted from the document, which leads to the problems of poor readability and non-representativeness of subject words. At present, the improved models include Character-Word topic model (*CWTM*), Dirichlet distribution-Word activation (*LDA – WAF*) combination model and Semantic Compression Based on Phase Topic Modeling (*SCPTM*) model based on *LDACOL*.

*CWTM* model introduces the relationship of feature words, puts a priori word on the basis of LDA modeling, and combines the relationship of subject words into the topic model, which improves the rationality of topic generation. *LDA – WAF* model associates the text topic with the document set, extracts the documents whose correlation degree is higher than a certain threshold to generate a new document, and then calculates the word excitation between words in the new

document through the word excitation model, thus obtaining a word activation matrix. Then, according to the word activation moment matrix to generate ordered topic words, the model has achieved good results in extracting document abstracts, but it ignores some useful information. *SCPTM* model mainly extracts the representative semantic words in the form of phrases, which is convenient for users to understand the document knowledge, and then uses the phrase mining model *LDACOL* to realize the topic modeling of document phrase. This model improves the problem of unstable topic assignment in traditional models, and makes the extracted topic words more in line with people's cognitive psychology. Therefore, *CWTM* model, *LDA – WAF* model and *SCPTM* model take more information into account in the modeling process, and integrate the context of words, thus improving the readability of the model results.

Dong et al. (2020) created a COVID-19 epidemic tracking map on January 22 to track and display real-time dynamic data of the epidemic situation in various countries based on China's official epidemic data. After that, the data source was expanded to the whole world, providing good support for everyone to grasp the epidemic's development status. In the article, we can see that the website can be produced quickly thanks to China's early and standardized data. At first, the map data was completely manually sorted and updated 4 to 5 times a day. Later, with the efforts of the R&D team, the data has gradually realized that it can be automatically updated every 20 minutes, supplemented by manual review, to ensure accuracy and improve the timeliness of data. It is not difficult to see that high-quality data provides the scientific research team with a basis for research and analysis and assistance in making relevant decisions. Moreover, more open, transparent, and accurate data are more convenient for scientific research institutions to collect and sort to make more important contributions to the development of epidemic control. At the same time, we need to organize and collect data faster and better to meet the epidemic's needs. These related maps were cited by mainstream media and health departments in many countries when they released epidemic data. As of April 9, the total number of visits to the epidemic map website reached 16 billion, with an average daily visit volume of 1 billion. This document and related websites provide a good idea for us to build our own website. We should ensure the data is updated in time to meet most of the needs. Moreover, we also need to open our website for free, and make our website more user-friendly, to better contribute to the fight against the epidemic.

## 2. Data

### 2.1. Data Collection

After a long search, we finally determined our data set. Recently, the Allen AI Institute, together with the Microsoft Research Institute, the National Institutes of Health (NIH) National Library of Medicine, and the White House Office of Science and Technology (OSTP), released the COVID-19 Open Research Data Set (COVID-19 Open Research Dataset, CORD-19), this data set provides free of charge more than 280,000 academic articles related to the entire family of coronaviruses, including more than 100,000 full-text data. The database will be updated daily with the latest research results, covers the latest research published in peer-reviewed publications and new content in archives such as bioRxiv and medRxiv. Simultaneously, this database is also completely free for researchers to perform natural language processing and data mining processing.

	cord_uid	sha	source_x	title	doi	pmcid	pubmed_id	license	abstract	publish_time
0	ug7v899j	d1aafb70c066a2068b02786f8929fd9c900897fb	PMC	Clinical features of culture-proven Mycoplasma...	10.1186/1471-2334-1-6	PMC35282	1.14726e+07	no-cc	OBJECTIVE: This retrospective chart review des...	2001-07-04
1	02tnwd4m	6b0567729c2143a66d737eb0a2f63f2dce2e5a7d	PMC	Nitric oxide: a pro-inflammatory mediator in l...	10.1186/rr14	PMC59543	1.1668e+07	no-cc	Inflammatory diseases of the respiratory tract...	2000-08-15
2	ejv2xin0	06ced00a5fc04215949aa72528f2eeaae1d58927	PMC	Surfactant protein-D and pulmonary host defense	10.1186/rr19	PMC59549	1.1668e+07	no-cc	Surfactant protein-D (SP-D) participates in th...	2000-08-25
3	2b73a28n	348055649b6b8cf2b9a376498df9bf41f7123605	PMC	Role of endothelin-1 in lung disease	10.1186/rr44	PMC59574	1.16869e+07	no-cc	Endothelin-1 (ET-1) is a 21 amino acid peptide...	2001-02-22
4	9785vg6d	5f48792a5fa08bed9f56016f4981ae2ca6031b32	PMC	Gene expression in epithelial cells in respons...	10.1186/rr61	PMC59580	1.16869e+07	no-cc	Respiratory syncytial virus (RSV) and pneumoni...	2001-05-11

**Figure 2.** Part of raw data

The standard format of documents in the database is the JSON format, which contains the following fields:

**paper\_id:** string format, the forty-digit SHA-1 hash value of the document in PDF format;

**metadata:** includes most of the content of the article, including the following fields:

- **title:** string format

- **author:** dictionaries of each author. The dictionary includes the author's first name (first), middle name (middle), last name (last), suffix (suffix), mailing address (affiliation), and email address (email).
- **abstract:** Abstract, each paragraph is a list, and each list contains the following information:
- **text:** body, string format
- **cite\_spans:** Cite related information. Each citation is a list. The list includes the number of characters at the beginning of the citation (start), the number of characters at the end (end), the serial number of the cited document (text), and the id ( ref\_id);
- **ref\_spans:** some inline references, the format is similar to cite\_spans;
- **section:** indicates the mark, fixed as Abstract

**body\_text:** body content, each paragraph is a list, similar to the abstract, it also includes the body (text), related information cited (cite\_spans), inline references (ref\_spans), and section tags (section, such as introduction, conclusion, etc.)

**bib\_entries:** related information of the citation, each reference is also saved in dictionary format, each dictionary contains the following fields:

- reference id (ref\_id, string format), document title (title, string format), author information (authors, similar to the previous format, dictionary format), year (year, integer), the publication (venue, string format), International Standard Serial Number (issn, string format), page number (pages, string format), some other id (other\_ids, including DOI and other information);

**ref\_entries:** inline information, each is stored as a dictionary, the dictionary includes content (text) and type (type, such as figure, table, etc.)

Our current research has downloaded nearly 100,000 articles in total, and our data set will be updated in real-time every week after the website is online. As shown in the figure 2, many articles were published before the outbreak of the 2019-nCoV epidemic, so we need to delete some old papers that have nothing to do with the 2019-nCoV. Selecting the English papers

published after January 2020 because the 2019-nCoV was highlighted after this time. The result as follows:

	cord_uid	sha	source_x	title	doi	pmcid	pubmed_id	license	abstract	publish_time
4290	f9tg6xsg	44449ad1cca160ce491d7624f8ae1028f3570c45	PMC	Dexmedetomidine improved renal function in pat...	10.1186/s40560-019-0415-z	PMC6939335	3.19088e+07	cc-by	BACKGROUND: Dexmedetomidine has been reported ...	2020-01-02
4291	f73c639r	def41c08c3cb1b3752bcff34d3aed7f8486e1c86	PMC	Aortic volume determines global end-diastolic ...	10.1186/s40635-019-0284-8	PMC6940405	3.18978e+07	cc-by	BACKGROUND: Global end-diastolic volume (GEDV)...	2020-01-02
4292	1qgpa45q	f5ae3f66face323615df39d839e056ab5fcc98df	PMC	Whole genome sequencing and phylogenetic analy...	10.1186/s12864-019-6400-z	PMC6941262	3.18985e+07	cc-by	BACKGROUND: Human metapneumovirus (HMPV) is an...	2020-01-02
4293	g34f5w6b	5be75ae4e7f8c892abd8dc396b9dbd035772c84a	PMC	European intensive care physicians' experience...	10.1186/s13756-019-0662-8	PMC6941296	3.19088e+07	cc-by	BACKGROUND: Antimicrobial resistance (AMR) com...	2020-01-02
4294	d1pd09zj	1cee4a0d0e823379ec34a462a04561bf4cd736a2	PMC	Synthetic carbohydrate-based vaccines: challen...	10.1186/s12929-019-0591-0	PMC6941340	3.19001e+07	cc-by	Glycoconjugate vaccines based on bacterial cap...	2020-01-03

**Figure 3. Raw Data After Selection**

## 2.2. Data preprocessing

Since 2018, Bert and its modified version have "slaughtered" a number of NLP tasks, leading to a great disturbance in the industrial and academic circles. However, Bert supports a maximum sequence length of 512. The length of a paper is so long that BERT can't handle it well. Therefore, we only extract the abstract of each article, which can summarize the content of the whole article, as the training corpus.

### Stop words

In general, stop words fall into two broad categories. One is the functional words contained in human language, which are so common that they have no practical meaning compared with other words, such as 'the', 'is', 'at', 'which', 'on', etc. Another kind words vocabulary words, such as 'want' and so on, these words, they are widely used for such a word search engine will be able to make the really relevant search results, there is no guarantee that it is difficult to help narrow your search, but also reduces the efficiency of search, so often put the word out of problems, so as to improve the search performance.

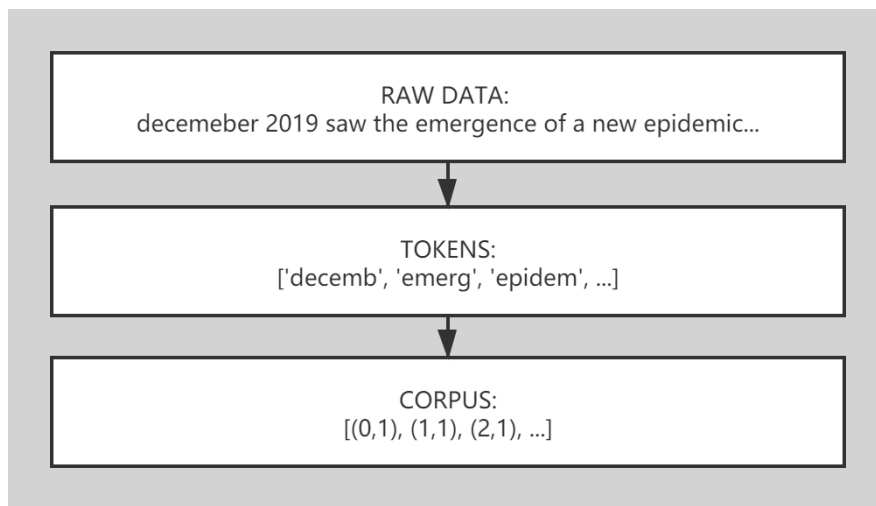
### Stemming

In NLP, we usually stem a sentence or document after word segmentation. Stemming is to remove the plural of some nouns, remove the different tenses of verbs and so on. The English

words obtained by word segmentation should be stemmed, which mainly includes changing the plural number of nouns into the singular number and changing other forms of verbs into basic forms.

## Result

Then we tokenize the sentences, remove punctuation, correct spelling, deal with stopping words and stemming the tokens. Finally, we get the clean training data and build relevant dictionaries and corpus. See the example as follows:



**Figure 4.** The process of data preprocessing

## 3. Models building

### 3.1 AutoEncoder

#### 3.1.1 Introduction of AutoEncoder

In the field of neural networks, people have long believed that multiple levels of nonlinear combination are the key. For example, in the case of modeling multiple variables with complex relations, identification of a variety of objects and so on, a lot of nonlinearities are required to achieve good generalization performance.

Let's start with a clear definition of a good representation, which means that a good representation helps people work on tasks that interest them. In other words, it will help the model

achieve a better performance on these tasks. Instead of saying we already know how to represent it and then we can improve the performance of the model.

For classification tasks, a "good" representation is one that helps the model become a better classifier based on objective criteria commonly used to evaluate algorithm performance.

For clustering tasks, a "good" representation is to help the model group together similar samples and spread out different samples.

However, if we want to learn from the recent breakthroughs in deep neural network training techniques, to get a good representation, we should not regard an error from a narrow classification task as the only or the main criterion of learning.

First of all, many related experiments have shown that ignoring specific classification problems and starting with the optimization of unsupervised learning criteria can actually greatly help people get "good" representation to improve the performance of subsequent classification tasks, and in the same way, it can also improve the effect of clustering.

Secondly, neural network is a discipline inspired by human neural network. The ability of human beings to acquire knowledge quickly in a new field is based on their understanding of old knowledge. This behavior is more like unsupervised learning behavior than supervised learning behavior.

The idea of autoencoder was first proposed in 1988. The model at that time was difficult to optimize due to its sparse data and high computational complexity, so it was not widely used. Until 2006, Hinton et al. used gradient descent to optimize RBM layer by layer, so as to realize the abstract representation of original samples / features, and achieved remarkable results in feature dimension reduction. This makes the use of neural network to build autoencoder method has been widely concerned.

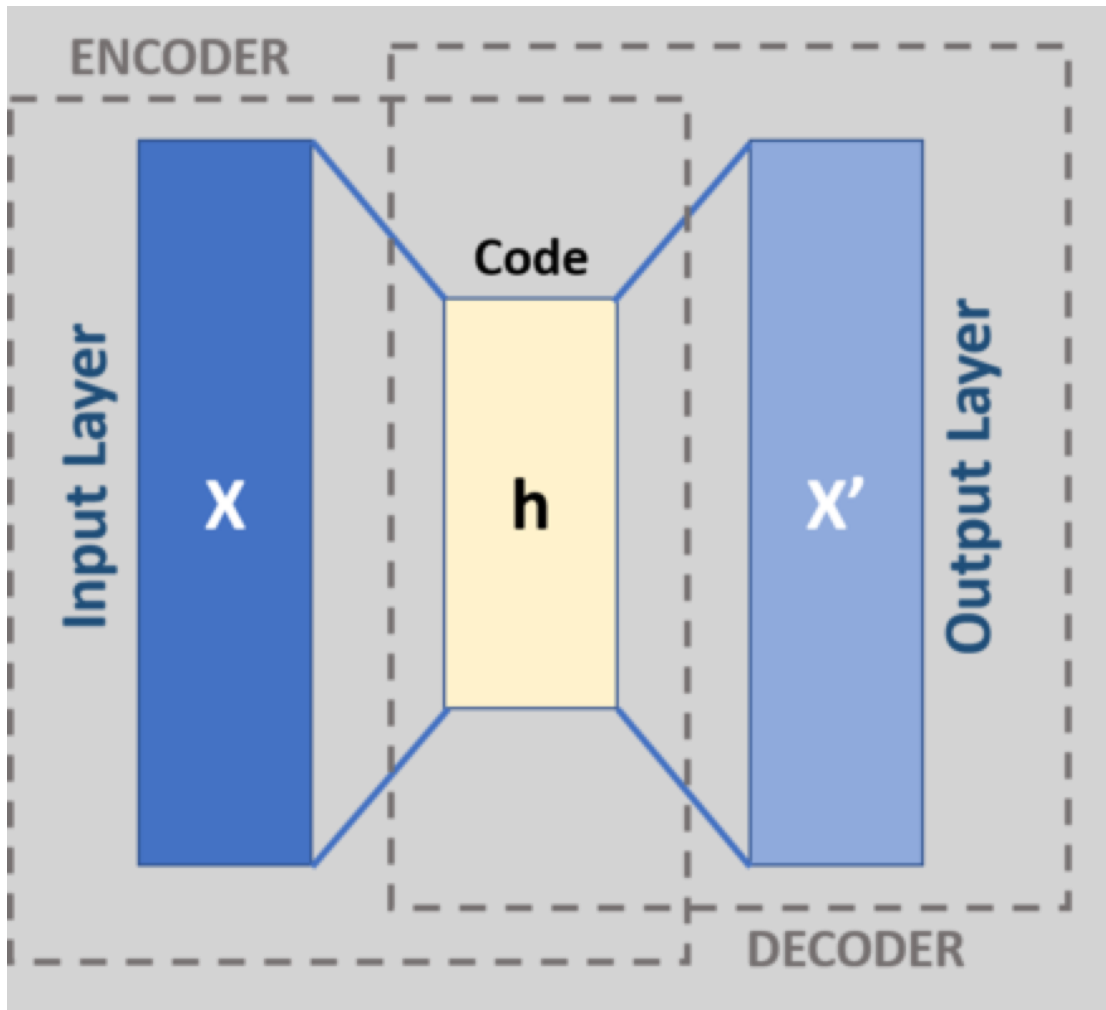
Autoencoder is an artificial neural network which uses unsupervised learning method to learn efficient data codings. The purpose of autoencoder is to extract and represent the features of high-dimensional data efficiently, and it is brilliant in the academic and industrial circles. It trains the network continuously through back propagation to ignore the signal noise. This model will



reconstruct the input and generate a simplified code as close to the original input as possible, which is the origin of its name. It is brilliant in the academic and industrial circles.

There are several variants of the basic model whose purpose is to force the learning representation of input to adopt useful attributes. Examples of regularized self encoders (sparse, de-noising and compressed self encoders), which demonstrate effective learning representations for subsequent classification tasks, and variational auto encoders, whose recent applications are used as generation models. Automatic encoder can solve many application problems, from face recognition to lexical semantic acquisition.

Before introducing the classic autoencoder model based on neural network, let's take a look at the basic idea of autoencoder framework as a whole, as shown in the figure below. Autoencoder framework includes two modules: encoding process and decoding process. The encoding process is to map the input samples  $X$  to a new feature space  $Z$  through our encoder ( $g$ ). Then, the features of this space  $Z$  are mapped back to the original space by a decoder( $f$ ) to get the reconstructed samples  $X'$ , which is the decoding process. In the whole process, our optimization goal is to make the reconstructed samples  $X'$  as close as possible to the original samples  $X$ , that is, to optimize the encoder and decoder by minimizing the error between  $X'$  and  $X$ , so as to learn a good representation of the input samples  $X$ .



**Figure 5** Frame of AutoEncoder

Here we can see that autoencoder does not need to use the label of the sample in the optimization process. In essence, the input of the sample is taken as the input and output of the neural network at the same time. By minimizing the reconstruction error, we hope to learn the abstract feature representation  $Z$  of the sample. This unsupervised optimization method greatly improves the universality of the model.

For the autoencoder model constructed by neural network, its encode part is to reduce the number of neurons in each layer to achieve the purpose of data dimension reduction and compression, while the decoding part, on the contrary, increases the number of neurons in each layer to increase the dimension of the abstracted features, and finally realize the reconstruction of input samples.

The attention here is that autoencoder learns the unique abstract representation of each sample through neural network, which brings a problem: when the parameters of neural network are complex to a certain extent, autoencoder is prone to the risk of over fitting.

### **3.1.2 Denoising AutoEncoder**

In order to alleviate the problem of over fitting of classical autoencoder, one method is to add random noise into the input; Vincent et al. [3] proposed denoising autoencoder, which adds random noise to the input layer of traditional autoencoder to enhance the robustness of the model. Another way is to combine the regularization idea. Rifai et al. [4] proposed the contractual autoencoder. By adding the Jacobian matrix normal form of the encoder into the autoencoder objective function, the encoder can learn the abstract features of anti-interference.

The following figure shows the model framework of denoising autoencoder. At present, there are two ways to add noise: adding random noise which obeys specific distribution; setting the specific proportion of input  $x$  to 0 randomly.

### **3.1.3 Sparse AutoEncoder**

In order to obtain the sparse high-dimensional abstract feature representation when learning the input sample representation, Ng et al. added a regularization term to control the sparseness in the original loss function. Sparse constraint can force only part of neurons in each layer of encoder to be activated, thus mapping samples into low dimensional sparse feature vectors.

Specifically, if the probability of activation of a single neuron is very small, then the network is considered to be sparse. Whether neurons are activated can be regarded as Bernoulli distribution of probability. Therefore, KL divergence can be used to measure the loss between the probability of neuron activation  $\hat{p}$  and the expected probability  $p$

### **3.1.4 Variational AutoEncoder**

Variational autoencoder (VAE) was proposed by Kingma et al. In 2014. The big difference of VAE is that VAE no longer maps input  $x$  to a fixed abstract feature  $Z$ , but assumes that the abstract feature  $Z$  of sample  $x$  obeys the normal distribution of  $(\mu, \sigma^2)$ , and then generates abstract feature  $Z$  through distribution. Finally, the output is obtained by the decoder based on  $Z$ . The model framework is shown in the figure below

Since the abstract feature  $Z$  is generated from the normal distribution sampling, the encoder part of VAE is a generation model, and then combined with the decoder to achieve reconstruction to ensure that information is not lost. VAE is a milestone research achievement, not because it is a good generation model, but mainly provides an idea of combining probability graph to enhance the robustness of the model. There are many VAE based extensions, including infovae, betaVAE and factorvae.

### **3.1.5 Adversarial AutoEncoder**

Since the generation model is introduced into autoencoder, it is necessary to introduce the idea of GAN into autoencoder, which has also achieved good results.

The network structure against self encoder is mainly divided into two parts: self coding part (upper part) and Gan discrimination network (lower part). The framework of the two is the combination of the two. The training process is divided into two stages: the first stage is the sample reconstruction stage, in which the parameters of the encoder and the coder are updated by gradient descent to minimize the reconstruction loss function; the second stage is the regularization constraint stage, in which the parameters of the discriminating network and the generating network (encoder part) are updated alternately to improve the ability of the encoder to confuse the discriminant network.

## **3.2 LDA**

LDA can be divided into the following five steps:

1. One function: Gamma function.
2. Four distributions: binomial distribution, multinomial distribution, beta distribution and Dirichlet distribution.
3. One concept and one idea: conjugate prior and Bayesian framework.
4. Two models: PLSA and LDA.
5. One sample: Gibbs sampling

### **3.2.1 Introduction of LDA**

According to the introduction on Wiki, LDA was proposed by BLEI, David M., ng, Andrew Y., and Jordan in 2003. It is a topic model. It can give the topic of each document in the document

set in the form of probability distribution. After extracting their topic (distribution) by analyzing some documents, LDA is a typical bag of words model, which means that a document is composed of a set of words, and each word has no order. Also, a document can have one or more topics generated for each word in the document. It is often used for topic clustering or text classification based on topic distribution.

How do humans generate documents? The three authors of LDA gave a simple example in the original paper. For example, suppose you have given these topics in advance: arts, budgets, children, education, and then get the words corresponding to each topic through training. As shown in the figure below:

<b>“Arts”</b>	<b>“Budgets”</b>	<b>“Children”</b>	<b>“Education”</b>
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

**Figure 6** Words in different Toics

Then select a certain topic with a certain probability, and then select a word under that topic with a certain probability, and repeat these two steps continuously. Finally, an article is generated as shown in the figure below (words with different colors correspond to words under different topics in the above figure)

The William Randolph Hearst Foundation will give \$1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be \$200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive \$400,000 each. The Juilliard School, where music and the performing arts are taught, will get \$250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual \$100,000 donation, too.

**Figure 7** A generated article

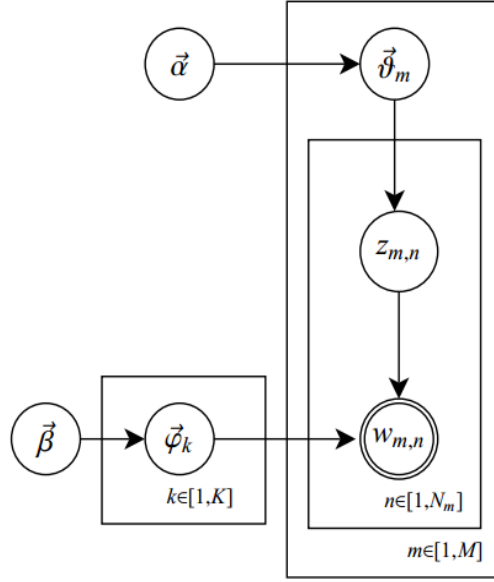
When we see an article, we often like to speculate on how the article is generated. We may think that the author first determines several themes of the article, and then uses words and sentences around these topics to express the writing.

So LDA is the reverse of this: according to a given document, its topic distribution is deduced.

Generally speaking, it can be assumed that human beings have written various articles according to the above document generation process. Now a small group of people want computers to use LDA to do one thing: your computer can infer and analyze what topics are written in each article on the network, and what is the probability size (topic distribution) of each topic in each chapter.

In LDA model, a document is generated as follows:

1. Generating topic distribution  $\theta_i$  of document  $i$  from Dirichlet distribution  $\alpha$
2.  $z_{i,j}$ , the topic of the  $j_{th}$  word of document  $i$  is generated by sampling from the polynomial distribution of topics  $\theta_i$
3. From the Dirichlet distribution  $\beta$ , the word distribution  $\phi_{i,j}$  corresponding to the topic  $z_{i,j}$  is generated.
4. Finally, words  $w_{i,j}$  are generated by sampling from the polynomial distribution  $\phi_{i,j}$  of words
5. The graph model structure of LDA is shown in the figure below:



**Figure 8** The structure of LDA

### 3.2.2 One function and Four distribution

#### Binomial distribution

Binomial distribution is advanced from Bernoulli distribution. Bernoulli distribution, also known as two-point distribution or 0-1 distribution, is a discrete random distribution. There are only two kinds of random variables in Bernoulli distribution. The binomial distribution, i.e. the Bernoulli test repeated  $N$  times, is recorded as  $X \sim b(n, p)$ . In short, the Bernoulli distribution is done once, and the binomial distribution is repeated  $N$  times. The probability density function of binomial distribution is as follows:

$$P(K = k) = \binom{n}{k} p^k (1 - p)^{n-k}$$

#### Multinomial distribution

It is the case that binomial distribution extends to multidimensional distribution. Multinomial distribution means that the values of random variables in a single experiment are no longer 0-1, but may have multiple discrete values  $(1, 2, 3, \dots, K)$ . For example, in the experiment of rolling dice with six faces, the results of  $N$  experiments obey the multinomial distribution of  $k = 6$ . Among them:

$$\sum_{i=1}^k p_i = 1, p_i > 0$$

### Beta distribution

Beta distribution is a density function of the conjugate prior distribution of Bernoulli distribution and binomial distribution.

The form as follows:

Given parameters  $\alpha > 0$  and  $\beta > 0$ , the value range is  $[0,1]$ .

$$f(x; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{(\beta-1)}$$

Where

$$\frac{1}{B(\alpha, \beta)} = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)}$$

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt = \frac{e^{-\gamma z}}{z} \prod_{k=1}^\infty \left(1 + \frac{z}{k}\right)^{-1} e^{z/k}, \quad \gamma \approx 0.577216$$

and  $\Gamma(z)$  is gamma function.

### Dirichlet distribution

Dirichlet distribution is the extension of beta distribution in high dimension. Its density function form is the same as that of beta distribution

$$f(x_1, x_2, \dots, x_k; \alpha_1, \alpha_2, \dots, \alpha_k) = \frac{1}{B(\alpha)} \prod_{i=1}^k x_i^{\alpha_i-1}$$

### 3.2.3 One concept and One idea

#### Conjugate Prior



In Bayesian statistics, if the posterior distribution and the prior distribution belong to the same category, the prior distribution and the posterior distribution are called the conjugate distribution, and the prior distribution is called the conjugate prior of the likelihood function.

### **Bayesian framework**

The fixed pattern of Bayesian thinking:

prior distribution  $\pi(\theta)$  + Sample information  $X$  = Posterior distribution  $\pi(\theta|X)$

The above thinking pattern means that the newly observed sample information will modify people's previous cognition of things. In other words, before getting the new sample information, people's cognition for  $\theta$  is a prior distribution  $\pi(\theta)$ , after getting the new sample information  $X$ , people's cognition to is  $\pi(\theta|X)$ .

By the way, the Frequentists and the Bayesians have different ways of thinking:

the Frequentists regards the parameter  $\theta$  to be inferred as a fixed unknown constant, that is, although the probability is unknown, it is at least a certain value. At the same time, the sample  $X$  is random. Therefore, the frequency school focuses on the sample space, and most of the probability calculation is aimed at the distribution of sample  $X$ ;

However, the Bayesians holds that the parameters  $\theta$  to be estimated are random variables and follow a certain distribution, while sample  $X$  is fixed. Because the samples are fixed, they focus on the distribution of parameters  $\theta$ .

### **3.2.4 Two models**

Before talking about LDA model, we should gradually understand the basic models: unigram model, mixture of unigrams model, and the PLSA model closest to LDA.

For the convenience of description, some variables are defined firstly:

1.  $w$  means word and  $V$  means the number of all words(fixed value).
2.  $z$  means topic and  $k$  means the number of topics(preset, fixed value).

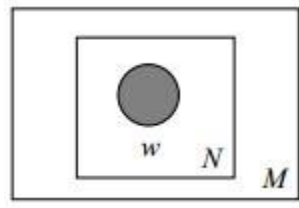
3.  $D = (W_1, W_2, \dots, W_M)$  means corpus and  $M$  means the number of documents in corpus(fixed value).
4.  $W = (w_1, w_2, \dots, w_N)$  means document and  $N$  means the number of words in a document(random variable).

### Unigram Model

For a document  $W = (w_1, w_2, \dots, w_N)$ , use  $p(w_n)$  represent the prior probability of the word  $w_n$ , the probability of generating documents  $w$  is:

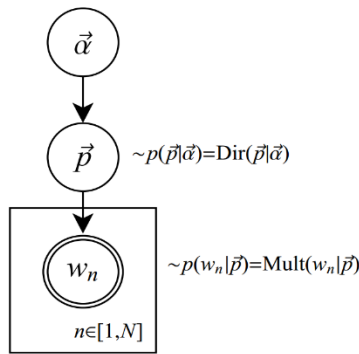
$$p(w) = \prod_{n=1}^N p(w_n)$$

The graph model is (the colored  $w$  represents the observable variable,  $N$  represents the total  $N$  words in a document, and  $M$  represents the  $M$  documentS):



**Figure 9** Diagram of Unigram model

or



**Figure 10** Flow of Unigram model

Unigram model assumes that the words in the text obey the multinomial distribution, and the prior distribution of multinomial distribution is Dirichlet distribution.

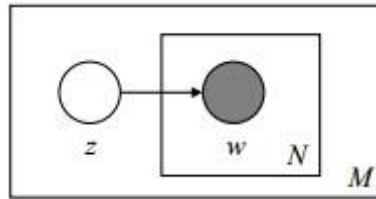
$w_n$  in the figure above indicates the  $n_{th}$  word observed in the text, and  $N \in [1, n]$  indicates that there are  $N$  words in the text. Add a box to indicate repetition, that is, there are  $N$  such random variables  $w_n$ .  $P$  and  $\alpha$  are hidden unknown variables where  $P$  is a parameter of the multinomial distribution to which a word follows and  $\alpha$  is a parameter of Dirichlet distribution (i.e. prior distribution of multinomial distribution). Generally,  $\alpha$  is given in advance by experience, and  $P$  is obtained by learning the words appearing in the observed text, which represents the probability of each word appearing in the text.

### Mixture of unigrams model

The generation process of Mixture of unigrams model is as follows: Select a topic  $z$  for a document, and then generate a document based on the topic. All the words in the document come from one topic. Suppose the topic has  $z_1, z_2, \dots, z_k$ , the probability of generating a document is:

$$p(w) = p(z_1) \prod_{n=1}^N p(w_n|z_1) + \dots + p(z_k) \prod_{n=1}^N p(w_n|z_k) = \sum_z p(z) \prod_{n=1}^N p(w_n|z)$$

The graph model is as follows (the colored  $w$  represents the observable variable, the uncolored  $z$  represents the unknown hidden variable,  $N$  represents the total  $N$  words in a document, and  $M$  represents the  $M$  documents).



**Figure 11** Diagram of Mixture of unigram model

### PLSA model

PLSA model is very close to LDA model. In fact, PLSA model plus a Bayesian framework can become an LDA model.

In the mixture of unigrams model above, we assume that a document has only one topic generated. In fact, an article often has multiple topics, but the probability of these multiple topics appearing in the document is different. For example, the documents introducing a country are often introduced from the perspectives of education, economy, transportation, etc. The process of document generation is as follows:

Suppose you have  $K$  optional topics and  $V$  optional words. Let's play a game of dice.

1. Suppose you make a "document topic" die with  $K$  faces (you can get any one of  $K$  topics by throwing this die) and  $K$  dice (each die corresponds to a topic,  $K$  dice corresponds to the previous  $K$  topics, and each face of the dice corresponds to the selected terms, and  $V$  faces corresponds to  $V$  optional words).
  - For example, you can make  $k = 3$ , that is, make a "document topic" dice with three themes, which can be education, economy and transportation. Then make  $v = 3$ , and make three dice with three sides of "theme word item". Among them, the words on the three sides of the dice of education theme can be: University, teacher, curriculum, and the words on the three aspects of the economic theme dice can be: market, enterprise, finance, transportation. The words on the three sides of the dice can be high-speed rail, automobile and airplane.
2. When writing a word, first roll the "document topic" die to select the topic. After getting the result of the topic, use the "topic word item" die corresponding to the topic result to select the word to write.
  - First, throw the "document topic" dice, and assume (with a certain probability) that the topic obtained is education, so the next step is to throw the education theme screen to get a word corresponding to the education theme screen (with a certain probability): University.
  - The process of rolling dice to generate words is simplified as follows: "first select the topic with a certain probability, and then select the word with a certain probability". In fact, at the beginning, there are three themes to choose from: education, economy and transportation. Why choose the theme of education? In fact, it is randomly selected, but this random follows a certain probability distribution. For example, the probability of selecting education theme is 0.5, the probability of

selecting economic theme is 0.3, and the probability of selecting traffic theme is 0.2, then the probability distribution of these three topics is {Education: 0.5, economy: 0.3, transportation: 0.2}. We call the probability distribution of each topic  $Z$  in document  $D$  as the topic distribution, and it is a multinomial distribution.

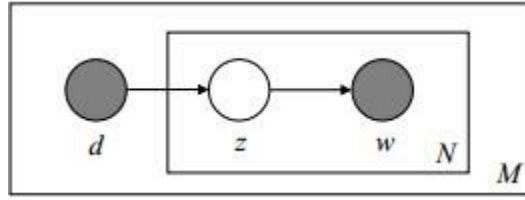
- Similarly, after randomly extracting the education theme from the topic distribution, we still face three words: University, teacher and course. These three words may be selected, but their probability of being selected is not the same. For example, the probability of university being selected is 0.5, the probability of teacher being selected is 0.3, and the probability of course being selected is 0.2. Then the probability distribution of these three words is {University: 0.5, teacher: 0.3, course: 0.2}. We call the probability distribution of each word  $w$  under the topic  $Z$  as the word distribution, which is also a multiple distribution.
  - Therefore, topic selection and word selection are two random processes. Firstly, the topic: education is extracted from the topic distribution {Education: 0.5, economy: 0.3, transportation: 0.2}, and then the word "University: 0.5, teacher: 0.3, course: 0.2" is extracted from the corresponding word distribution {University: 0.5, teacher: 0.3, course: 0.2}.
3. Finally, you repeatedly throw the "document subject" dice and "subject word" dice repeatedly, repeat  $n$  times (generate  $n$  words), complete a document, and repeat the method of generating a document  $m$  times, then complete the  $m$  document.

The above process is abstracted out as PLSA document generation model. In this process, we do not pay attention to the word and word order, so PLSA is a word bag of words method. The whole process of document generation is to select the topic of document generation and determine the topic generation words.

Conversely, since the document has already been generated, how to deduce the theme of the generated good document? The purpose of topic modeling is to automatically discover the topic (distribution) in the document set, which is the reverse process of generating the document.

In other words, human beings write all kinds of articles according to the document generation model, and then throw them to the computer, which is equivalent to that the computer sees the articles that have been written. Now the computer needs to sum up the theme of an article according to a series of words seen in an article, and then get the probability of each topic's occurrence: topic distribution. That is, document  $d$  and word  $w$  are observable, but topic  $z$  is hidden.

As shown in the figure below (colored  $d$  and  $w$  represent observable variables, uncolored  $z$  represents unknown hidden variables,  $N$  represents a total of  $N$  words in a document, and  $M$  represents  $M$  documents):



**Figure 12** diagram of pLSA

In the figure above, document  $d$  and word  $w$  are the samples we get (the samples are random, and the parameters are unknown but fixed, so PLSA belongs to the Frequentists thought. It is different from LDA to be introduced below: the sample is fixed, the parameter is unknown but not fixed, it is a random variable, obeying a certain distribution, so LDA belongs to the Bayesians thought), observable, so  $p(w_j|d_i)$  is known for any document.

Thus, according to a large number of known document-word information  $p(w_j|d_i)$ , document-topic  $p(z_k|d_i)$  and topic-word  $p(w_j|z_k)$  can be trained, as shown in the following formula:

$$p(w_j|d_i) = \sum_{k=1} p(w_j|z_k) p(z_k|d_i)$$

Therefore, the generation probability of each word in the document is obtained as follows:

$$p(d_i, w_j) = p(d_i)p(w_j|d_i) = p(d_i) \sum_{k=1}^K P(w_j|z_k)p(z_k|d_i)$$

Since  $p(d_i)$  can be calculated in advance, while  $P(w_j|z_k)$  and  $p(z_k|d_i)$  are unknown,  $\theta = (P(w_j|z_k), p(z_k|d_i))$  is the parameter (value) we want to estimate. In general, it is to maximize the  $\theta$ .

The commonly used parameter estimation methods include maximum likelihood (MLE), maximum a posteriori (MAP), Bayesian estimation and so on. Because the parameter to be estimated contains hidden variable  $Z$ , we can consider EM algorithm.

### **Introduction of EM algorithm**

EM algorithm, known as expectation maximization algorithm, is expected maximum algorithm.

The research on EM algorithm originates from the problem of statistical error analysis. In 1886, American mathematician Simon Newcomb proposed an iterative solution technique similar to EM algorithm when using Gaussian mixture model (GMM) to explain the long tail effect of observation error. After the emergence of maximum likelihood estimation (MLE), many theories and methods have been proposed. In 1926, British scholar Anderson McKendrick put forward the new comb theory applied to medical samples. In 1956, Michael Healy and Michael Westmacott proposed an iterative method to estimate missing data in statistical experiments, which is considered as a special case of EM algorithm. In 1970, B. J. n. light used MLE to discuss the type I censored data of exponential family distribution. From 1971 to 1974, the maximum likelihood estimation of exponential family distribution samples has been greatly developed. Rolf sundberg gave a complete derivation of iterative calculation during this period.

EM algorithm was formally proposed by American mathematicians Arthur Dempster, Nan laird and Donald Rubin. Their research published in 1977 summarized the previous EM algorithm as a special case and gave the calculation steps of the standard algorithm. The EM algorithm is also known as Dempster Laird Rubin algorithm. In 1983, American mathematician c.f. Jeff Wu gave the proof of the convergence of EM algorithm outside the exponential family distribution.

It is frequently used to deal with missing data and observe unidentified variables, e.g., in pricing and managing risk of a portfolio, for censored data commonly encountered in survival analysis, for data clustering in machine learning and computer vision, for parameter estimation of mixed models, notably in quantitative genetics, in medical image reconstruction, in structural engineering and so on.

The basic idea is: first, select a value randomly to initialize the value  $\theta^0$  to be estimated, and then iterate to find a better  $\theta^{n+1}$ , so that the likelihood function  $L(\theta^{n+1})$  is larger than the original  $L(\theta^n)$ . In other words, suppose you get  $\theta^n$  now and want to find  $\theta^{n+1}$ .

The general EM algorithm as follows:

Denote the observed data by  $y$ , and the missing values by  $u$ .

Denote the joint densities of  $y$  and  $(y, u)$  by  $f(y; \theta)$  and  $f(y, u; \theta)$ , respectively; the conditional density of  $u$  given  $y$  by  $f(u | y, \theta)$ .

By using Bayes Theorem it is straightforward to show that

$$\begin{aligned} \log f(y, u; \theta) &= \log f(y; \theta) + \log f(u | y, \theta) \\ \Rightarrow l(\theta, y, u) &= l(\theta, y) + \log f(u | y, \theta) \end{aligned}$$

where  $l(\theta, y, u)$  and  $l(\theta, y)$  are the log-likelihoods of  $(y, u)$  and  $y$ , respectively.

The EM algorithm is based on maximizing approximations of  $l(\theta, y, u)$  based on the observation  $y$ .

Of course, in reality,  $l(\theta, y, u)$  is unknown because  $u$  is unobserved. However, let us consider the conditional expectation of  $l(\theta, y, U)$ , given what we observe  $y$ , as a function of  $y$ . That is,

$$\begin{aligned} Q(\theta_0, \theta, y) &= E_{\theta_0} [l(\theta, y, U) | y] \\ &= \int l(\theta, y, u) f(u | y, \theta_0) du \end{aligned}$$



where  $\theta_0$  is the initial or pre-estimated value for  $\theta$ .

It is worth noting that  $Q(\theta_0, \theta, y)$  can be viewed as the best predictor of the complete likelihood  $l(\theta, y, u)$  given  $(y, \theta_0)$ . In other words,  $u$  in  $l(\theta, y, u)$  is estimated by minimizing the posterior MSE:

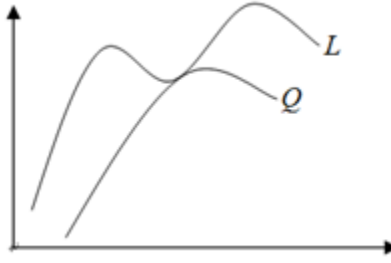
$$\hat{u} = E_{\theta_0}(U|Y) = \underset{g}{\operatorname{argmin}} E_{\theta_0}[U - g(Y)]^2$$

The EM algorithm is based on iterating  $Q(\cdot)$  in such a way that at each iteration we obtaining an estimator which gives a larger value of  $Q(\cdot)$  (also a larger  $l(\theta, y)$ ). We describe the EM algorithm below.

Initialization: Define an initial value  $\theta_0 \in \Omega$  (not  $u^0$ , and needn't have an explicit value(s) or functional form).

- Iteration 1: (*E – Step*) Evaluate  $Q(\theta_0, \theta, y)$ .  
                   (*M – Step*) Find  $\theta_1 = \operatorname{argmax}_{\theta} Q(\theta_0, \theta, y)$
- Iteration  $k+1$ : Evaluate (*E – Step*)  $Q(\theta_k, \theta, y)$ , and find (*M – Step*)  $\theta_{k+1} = \operatorname{argmin}_{\theta} Q(\theta_k, \theta, y)$ ,  $k = 1, 2, \dots$
- Iteration stops when  $\theta_k$  and  $\theta_{k+1}$  are sufficiently close to each other and set  $\hat{\theta} = \theta_{k+1}$  as the final estimator for  $\theta$ .

The above process is like that in a two-dimensional plane, there are two disjoint curves, one is on the upper (referred to as the upper curve  $L$ ), one is on the lower (referred to as the lower curve  $Q$ ). Now the upper curve is unknown, only the lower curve is known. In order to solve the highest point of the upper curve, we try to increase the lower curve to make the lower curve constantly approach the upper curve. The lower curve reaches the local maximum value at a certain point and is equal to the value of the upper curve at this point. Record this value, and then continue to increase the lower curve to find the same value on the lower curve and the upper curve and the iteration stops until  $L(\theta)$  convergence (i.e.  $Q$  convergence), Thus, the local maximum value on the current lower curve is used as the global maximum value of the upper curve (in other words, the EM algorithm does not guarantee to find the global optimal value). As shown in the figure below:



**Figure 13** the Example of EM Algorithm

Thus, in pLSA:

1. Since  $P(w_j|z_k)$  and  $p(z_k|d_i)$  are unknown, we use EM algorithm to estimate the value of  $\theta = (P(w_j|z_k), p(z_k|d_i))$ .
2. Then,  $\phi_{k,j}$  is used to represent the probability that the word  $w_j$  appears in topic  $z_k$ , that is,  $\theta_{i,k}$  is used to represent the probability that topic  $z_k$  appears in document  $d_i$ . thus,  $P(w_j|z_k)$  is transformed into a "topic-word" matrix  $\Phi$  (topic generated words), and  $p(z_k|d_i)$  is transformed into a "document-topic" matrix  $\theta$  (document generated topic).
3. Finally,  $\phi_{k,j}, \theta_{i,k}$  are solved.

## LDA Model

In fact, if you understand the PLSA model, you can almost understand the LDA model as soon as possible, because LDA just use a Bayesian framework on the basis of PLSA, that is, LDA is the Bayesian version of PLSA (because LDA is Bayesian, we need to consider historical prior knowledge and add two prior parameters).

### Comparison of PLSA and LDA

In the PLSA model, we get the "document-word" generation model according to the following steps:

1. Select a document  $d_i$  according to the probability  $p(d_i)$
2. After selecting document  $d_i$ , determine the topic distribution of the article
3. An implicit topic category  $p(z_k|d_i)$  is selected from the topic distribution according to the probability  $z_k$

4. After selecting  $z_k$ , determine the word distribution under the topic
5. Choose a word  $w_j$  "from word distribution according to probability  $P(w_j|z_k)$ "
6. For LDA:
7. Select a document  $d_i$  according to the prior probability  $p(d_i)$
8. The topic distribution  $\theta_i$  of document  $d_i$  is generated by sampling from Dirichlet distribution  $\alpha$ , in other words, the topic distribution  $\theta_i$  is generated by the Dirichlet distribution with  $\alpha$  as the super parameter
9. The topic  $z_{i,j}$  of the  $j_{th}$  word of the document  $d_i$  is generated by sampling from the polynomial distribution  $\theta_i$  of the topic
10. The word distribution  $\phi_{z_{i,j}}$  of topic  $z_{i,j}$  is generated by sampling from Dirichlet distribution  $\beta$ , in other words, the word distribution  $\phi_{z_{i,j}}$  is generated by the Dirichlet distribution with  $\beta$  as the super parameter
11. Sampling from the polynomial distribution  $\phi_{z_{i,j}}$  of words, the word  $w_{i,j}$  "is finally generated
12. From the above two processes, LDA adds two Dirichlet priors to the theme distribution and word distribution on the basis of PLSA.

In pLSA, after the topic distribution and word distribution are determined, the specific topic and word item are selected with a certain probability ( $p(z_k|d_i), P(w_j|z_k)$ ) to generate a good document. Then, when the topic distribution and word distribution of the generated document are deduced, the EM algorithm (maximum likelihood estimation) is used to solve the unknown but fixed parameters:  $\phi_{k,j}$  (converted from  $P(w_j|z_k)$ ) and  $\theta_{i,k}$  (converted from  $p(z_k|d_i)$ ).

However, in LDA which are under Bayesian framework, we no longer think that topic distribution (probability distribution of each topic appearing in a document) and word distribution (probability distribution of words appearing in a certain topic) are unique (but random variables), but there are many possibilities. But a document must correspond to a topic distribution and a word distribution. What should we do? LDA gets two Dirichlet prior parameters for them. The Dirichlet priors extract a topic distribution and word distribution randomly for a document.

Therefore, the essential difference between PLSA and LDA is that they use different ideas to estimate unknown parameters. The former uses the frequentists thought, and the latter uses the Bayesians thought.

### 3.2.5 Gibbs sampling

Gibbs sampling is a special Markov chain algorithm, which is often used to solve a series of problems including matrix decomposition, tensor decomposition and so on. The word "alternate" means that Gibbs sampling is an iterative algorithm, and the corresponding variables will be used alternately in the process of iteration. In addition, the word "condition" is added because the core of Gibbs sampling is Bayesian theory, which is based on the prior knowledge and observation data and takes the observation value as the condition to infer the posterior distribution.

In LDA, according to Bayes rule and Dirichlet prior, we can calculate the posterior distribution of topics on each document and the posterior distribution of words under each topic (as we can see above: the posterior distribution is the Dirichlet distribution just like their prior distribution). The problem of unknown subject distribution and word distribution is solved successfully.

### 3.2.6 LDA+BERT+AutoEncoder+K-Means

There are many methods to combine the vector from LDA and the vector from BERT I have tried.

The version 1 is using the LDA to extract the probability vector of which cluster the text belongs to, using the BERT to get the embedding of the whole abstract, and then multiply the LDA vector with a parameter, and concatenate them with the Bert vector. Finally, using the Autoencoder to reduce the vector's dimension to get the final vector.

Code:

The vector of lda:

```
1. def get_vec_lda(model, corpus, k):
2.     n_doc = len(corpus)
3.     vec_lda = np.zeros((n_doc, k))
4.     for i in range(n_doc):
5.         for topic, prob in model.get_document_topics(corpus[i]):
```

```

6. vec_lda[i, topic] = prob
7. return vec_lda

```

The final vector:

```

1. vec_lda = self.vectorize(sentences, token_lists, method='LDA')
2. vec_bert = self.vectorize(sentences, token_lists, method='BERT')
3. vec_ldabert = np.c_[vec_lda * self.gamma, vec_bert]
4. self.vec['LDA_BERT_FULL'] = vec_ldabert
5. if not self.AE:
6.     self.AE = Autoencoder()
7.     self.AE.fit(vec_ldabert)
8.     vec = self.AE.encoder.predict(vec_ldabert)

```

The version 2 is using the LDA to extract the probability vector of which cluster the text belongs to, saving the keywords for each topic and using BERT to convert them into vectors, concatenate them as the vector from LDA. Using the BERT to get the embedding of the whole abstract, and then multiply the LDA vector with a parameter, and concatenate them with the Bert vector. Finally, using the Autoencoder to reduce the vector's dimension to get the final vector.

code:

The vector of lda:

```

1. def get_vec_lda(model, corpus, k):
2.     n_doc = len(corpus)
3.     mylabels = []
4.     for i in range(n_doc):
5.         mylabels.append(max(model.get_document_topics(corpus[i]), key=lambda x: x[1])[0])
6.     mylabels = np.array(mylabels)
7.     lda_topic_words = get_topic_words(token_lists, mylabels)
8.     lda_topic_sentences = []
9.     for i in range(len(lda_topic_words)):
10.        lda_topic_sentences.append(' '.join(lda_topic_words[i]))
11.    vec_topic_bert = vectorize(lda_topic_sentences, token_lists, method='BERT')
12.    vec_lda = np.zeros((n_doc, 768+k))#k
13.    for i in range(n_doc):
14.        for topic, prob in model.get_document_topics(corpus[i]):
15.            vec_lda[i, topic] = prob
16.            vec_lda[i] = np.hstack((vec_lda[i][:k], vec_topic_bert[vec_lda[i].argmax()]))
17.    return vec_lda

```

The final vector:

```

1. vec_lda = self.vectorize(sentences, token_lists, method='LDA')
2. vec_bert = self.vectorize(sentences, token_lists, method='BERT')
3. vec_ldabert = np.c_[vec_lda * self.gamma, vec_bert]
4. self.vec['LDA_BERT_FULL'] = vec_ldabert
5. if not self.AE:

```

```

6.     self.AE = Autoencoder()
7.     self.AE.fit(vec_ldabert)
8.     vec = self.AE.encoder.predict(vec_ldabert)

```

The version 3 is using the LDA to extract the probability vector of which cluster the text belongs to, saving the keywords for each topic and using BERT to convert them into vectors, concatenate them as the vector from LDA. Using the Autoencoder to reduce the vector's dimension. Using the BERT to get the embedding of the whole abstract. And then multiply the LDA vector with a parameter, concatenate them with the Bert vector. Finally, using the Autoencoder to reduce the vector's dimension to get the final vector.

code:

The vector of lda:

```

1. def get_vec_lda(model, corpus, k):
2.     n_doc = len(corpus)
3.     mylabels = []
4.     for i in range(n_doc):
5.         mylabels.append(max(model.get_document_topics(corpus[i]), key=lambda x: x[1])[0])
6.     mylabels = np.array(mylabels)
7.     lda_topic_words = get_topic_words(token_lists, mylabels)
8.     lda_topic_sentences = []
9.     for i in range(len(lda_topic_words)):
10.        lda_topic_sentences.append(' '.join(lda_topic_words[i]))
11.    vec_topic_bert = vectorize(lda_topic_sentences, token_lists, method='BERT')
12.
13.    vec_lda = np.zeros((n_doc, 768+k))#k
14.    for i in range(n_doc):
15.        for topic, prob in model.get_document_topics(corpus[i]):
16.            vec_lda[i, topic] = prob
17.            vec_lda[i] = np.hstack((vec_lda[i][:k], vec_topic_bert[vec_lda[i].argmax()]))
18.    return vec_lda

```

The final vector:

```

1. vec={}
2. vec_lda = self.vectorize(sentences, token_lists, method='LDA')
3. if not self.AE:
4.     self.AE = Autoencoder(latent_dim=64)
5.     self.AE.fit(vec_lda)
6.     vec_lda = self.AE.encoder.predict(vec_lda)
7.     self.vec_lda = vec_lda
8. vec_bert = self.vectorize(sentences, token_lists, method='BERT')
9. self.vec_bert = vec_bert
10. vec_ldabert = np.c_[vec_lda * self.gamma, vec_bert]
11. self.vec['LDA_BERT_FULL'] = vec_ldabert
12. self.AE = Autoencoder(latent_dim=64)
13. self.AE.fit(vec_ldabert)
14. vec = self.AE.encoder.predict(vec_ldabert)

```

The version 4 is using the LDA to extract the probability vector of which cluster the text belongs to, saving the keywords for each topic and using BERT to convert them into vectors, concatenate them as the vector from LDA. Using the Autoencoder to reduce the vector's dimension. Using the BERT to get the embedding of the whole abstract and using the Autoencoder to reduce the vector's dimension. Finally, Add two vectors together as the final vector.

code:

The vector of lda:

```

1. def get_vec_lda(model, corpus, k):
2.     n_doc = len(corpus)
3.     mylabels = []
4.     for i in range(n_doc):
5.         mylabels.append(max(model.get_document_topics(corpus[i]), key=lambda x: x[1])[0
6.         ])
7.     mylabels = np.array(mylabels)
8.     lda_topic_words = get_topic_words(token_lists, mylabels)
9.
10.    lda_topic_sentences = []
11.    for i in range(len(lda_topic_words)):
12.        lda_topic_sentences.append(' '.join(lda_topic_words[i]))
13.    vec_topic_bert = vectorize(lda_topic_sentences, token_lists, method='BERT')
14.    vec_lda = np.zeros((n_doc, 768+k))#k
15.    for i in range(n_doc):
16.        for topic, prob in model.get_document_topics(corpus[i]):
17.            vec_lda[i, topic] = prob
18.        vec_lda[i] = np.hstack((vec_lda[i][:k], vec_topic_bert[vec_lda[i].argmax()]))
19.    return vec_lda

```

The final vector:

```

1. vec={}
2. vec_lda = self.vectorize(sentences, token_lists, method='LDA')
3. if not self.AE:
4.     self.AE = Autoencoder(latent_dim=128)
5.     self.AE.fit(vec_lda)
6.     vec_lda = self.AE.encoder.predict(vec_lda)
7.     self.vec_lda = vec_lda
8. vec_bert = self.vectorize(sentences, token_lists, method='BERT')
9. self.AE = Autoencoder(latent_dim=128)
10. self.AE.fit(vec_bert)
11. vec_bert = self.AE.encoder.predict(vec_bert)
12. self.vec_bert = vec_bert
13. vec_ldabert = vec_lda + vec_bert
14. vec = vec_ldabert

```

The version 5 is using the LDA to extract the probability vector of which cluster the text belongs to, saving the keywords for each topic and using BERT to convert them into vectors, concatenate them as the vector from LDA. Using the Autoencoder to reduce the vector's dimension. Using the BERT to get the embedding of the whole abstract and using the Autoencoder to reduce the vector's dimension. Finally, multiply the LDA vector with a parameter, concatenate them with the Bert vector as the final vector.

code:

The vector of lda:

```
1. def get_vec_lda(model, corpus, k):
2.     n_doc = len(corpus)
3.     mylabels = []
4.     for i in range(n_doc):
5.         mylabels.append(max(model.get_document_topics(corpus[i]), key=lambda x: x[1])[0])
6.     mylabels = np.array(mylabels)
7.
8.     lda_topic_words = get_topic_words(token_lists, mylabels)
9.
10.    lda_topic_sentences = []
11.    for i in range(len(lda_topic_words)):
12.        lda_topic_sentences.append(' '.join(lda_topic_words[i]))
13.    vec_topic_bert = vectorize(lda_topic_sentences, token_lists, method='BERT')
14.    vec_lda = np.zeros((n_doc, 768+k))#k
15.    for i in range(n_doc):
16.        for topic, prob in model.get_document_topics(corpus[i]):
17.            vec_lda[i, topic] = prob
18.            vec_lda[i] = np.hstack((vec_lda[i][:k], vec_topic_bert[vec_lda[i].argmax()])))
19.    return vec_lda
```

The final vector:

```
1. vec={}
2. vec_lda = self.vectorize(sentences, token_lists, method='LDA')
3. if not self.AE:
4.     self.AE = Autoencoder(latent_dim=64)
5.     self.AE.fit(vec_lda)
6.     vec_lda = self.AE.encoder.predict(vec_lda)
7.     self.vec_lda = vec_lda
8.     print(len(vec_lda[0]))
9. vec_bert = self.vectorize(sentences, token_lists, method='BERT')
10. self.AE = Autoencoder(latent_dim=128)
11. self.AE.fit(vec_bert)
12. vec_bert = self.AE.encoder.predict(vec_bert)
13. self.vec_bert = vec_bert
14. vec_ldabert = np.c_[vec_lda * self.gamma, vec_bert]
15. vec = vec_ldabert
```

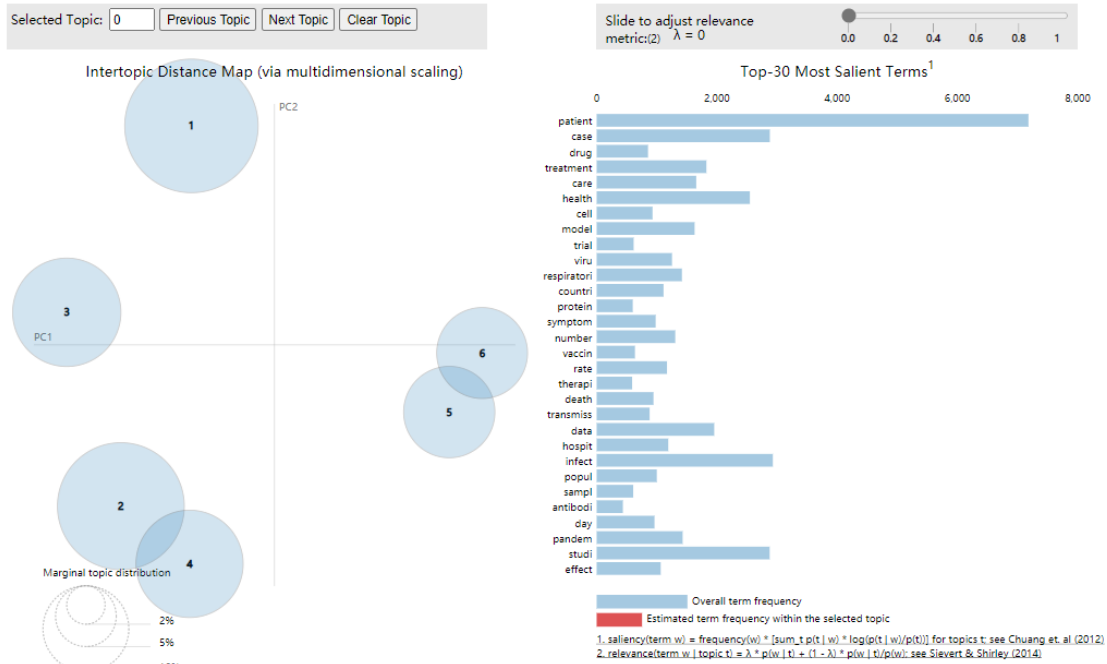


After getting the vector, clustering method is used to cluster these vectors. At present, we only use k-means to cluster articles.

### 3.2.7 Visualization

#### Topic Modeling

Selecting the number of topics as 5 for topic modelling.

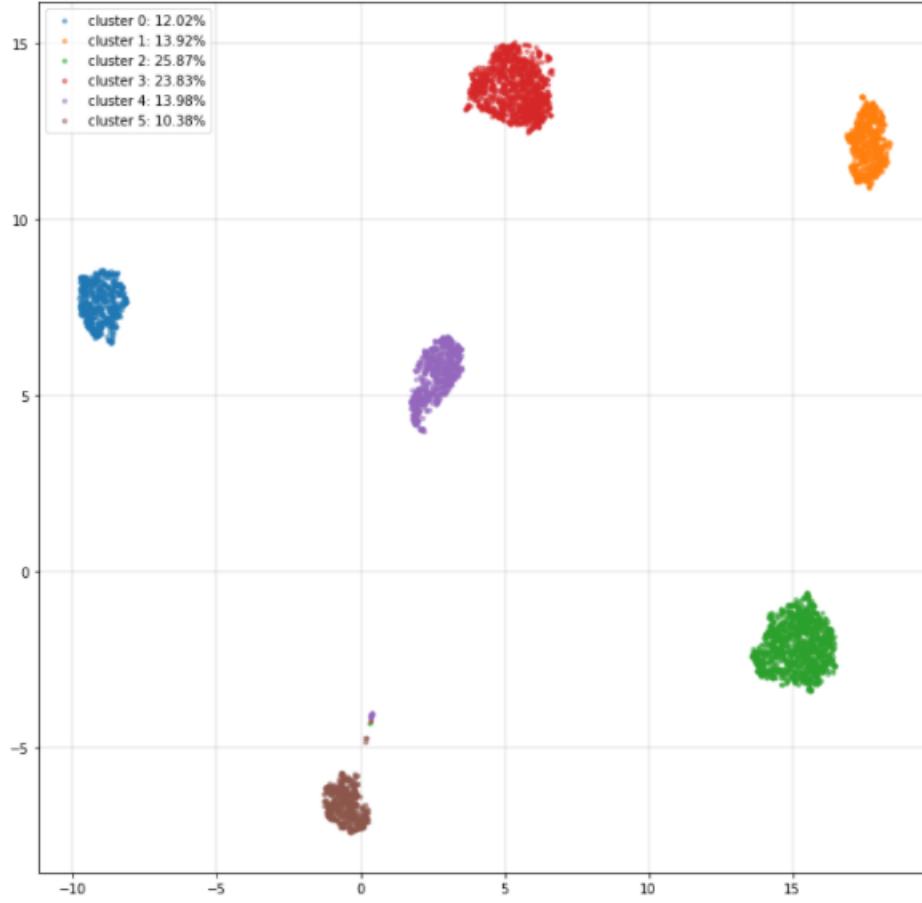


**Figure 14. The visualization of LDA**

The overlapping of some clusters in the above figure does not mean that the partition between them is not good, because the final variable is high-dimensional, and the visualization is only two-dimensional. Therefore, when the dimensionality is reduced for visualization, there will inevitably be some overlap.

#### LDA+BERT+K-Means

Using the version 5 method to get the final vector. At present, we only use k-means to cluster articles.



**Figure 15.** The visualization of LDA+BERT+K-Means

### 3.2.8 Evaluation

Topic coherence is a measure to evaluate the result of a topic model. Each generated topic is composed of words. Topic coherence is defined as the average / median of word pair similarity (such as PMI) in the topic, which is applied to the Top N words of the topic. A good model has a high coherence score, which means that it can generate coherent topics.

**Table 1.** The coherence of two model

LDA	LDA+BER T+K-Means v1	LDA+BER T+K-Means v2	LDA+BER T+K-Means v3	LDA+BER T+K-Means v4	LDA+BERT +K-Means v5
<b>0.487</b>	<b>0.477</b>	<b>0.488</b>	<b>0.492</b>	<b>0.508</b>	<b>0.515</b>

### 3.2.9 Conclusion

From the results, the effect of the latter two methods will be better. I guess the reason is that LDA variable and Bert variable use autoencoder separately for dimension reduction, instead of mixing them into the same autoencoder for training as the previous methods. The LDA variable is the summary of the whole article, and the Bert variable is the representation of each word, which obtains the local characteristics of the article. Therefore, it is better to merge them in the final stage rather than continue to use other methods after the combination.

### **3.2.10 Future Work**

As for the future work, I think there are many improvements in LDA model and Bert model. As mentioned in the literature review, LDA model has many improvements, such as focusing on the order of words, using tags to expand the model and so on. For the part of the Bert model, if possible, we can use more relevant papers, combined with the ideas of existing improved Bert models, such as xlnet, Roberta and so on, to train a model more suitable for the new coronavirus, so as to obtain better embedding. As for the clustering method, in the LDA + Bert model of this paper, due to the time limit, I put most of my energy on how to get better vectors, less attention on how to improve the clustering model to obtain better clustering effect. I will spend more time in the field of NLP clustering.

## **3.3 BERT**

The traditional word vector representation method is based on the N-Gram model and uses Word2Vec for semantic vector representation. Compared with the original One-Hot form, it solves the high latitude, sparse vector and related words of the One-Hot form. Can not reflect the characteristics of its semantic similarity. In actual language scenarios, each word has its own expression in different contexts. Different upper and lower sentences correspond to different vector representations. For each sentence, contextual semantics should be linked to the meaning of the specific scenario. Instead of using word vectors for weighting or arithmetic average to calculate its vector representation.

BERT, named Bidirectional Encoder Representations from Transformers, is new language representation model put forward by Google in 2018 and has made outstanding performance in many NLP tasks.

The model is divided into two sections, Encoder and Decoder. For encoder, there are two layers, a self-attention layer and a feedforward neural network. Self-attention can help the current node focus on more than the current word, thus obtaining the semantic meaning of the context. Decoder also includes the two layers as Encoder, but in between the two layers is a layer of attention, which helps the current node get the key content that needs to be paid attention at present.

BERT uses the model of Transformer Encoder as the language model [6], and replaces the structure such as RNN, CNN, but adopts the attention mechanism to calculate the relationship between input and output. With a self-attention mechanism in Transformer Encoder, BERT comes with a bidirectional function.

In order to obtain higher-level semantic representation of Sentence level than words, BERT added Next Sentence to perform joint training together with Masked-LM, and conducted training on a large number of corpus data to obtain a pre-training model. The Masked Language Model needs to predict the masked word, and Next Sentence Prediction is for the model to understand the relationship between sentences.. BERT used Wikipedia text for pre-training during development, so BERT has a wide range of text processing capabilities.

In order to adapt to the transfer learning under multi-tasks, BERT designed more general input and output layers.

BERT sentence vector representation is an important part of the model. First of all, the BERT selected in this article is a 24-layer Transformer for English trained by Google using a 1 billion corpus and contains 1024 hidden layer units. Clustering analysis and practice of research scholars have shown that the penultimate layer of the network has good results. Therefore, the parameters of the model are set to extract the corresponding number of layers, and the open source BERT script is used to build local services. By loading the pre-trained model, all sentences in the article are divided according to periods, and the divided sentences are input as independent sentences into the BERT model, and finally a sentence vector with 768 dimensions for each sentence is obtained. The sentence vectors extracted through the BERT pre-training model have richer semantic understanding capabilities and deep semantic representation capabilities. It has an indisputable advantage for judging the degree of text similarity and attribution in subsequent modelling.

In short, BERT generates sentence vectors with the advantage of understanding sentence meaning and eliminating the error caused by word vector weighting.

We use the pre-trained training model to extract feature vectors from the input text. In the feature extraction model, the best result is the penultimate layer, because the value of the last layer is too close to the target, while the value of the first several layers may not have fully learned semantic. We choose the penultimate layer as the output result.

### **BERT part code explanation**

The first is the configuration of BERT. The model configuration is relatively simple, in order: dictionary size, number of hidden layer neurons, number of transformer layers, number of attention heads, activation function, number of middle layer neurons, hidden layer dropout ratio, attention dropout ratio, sequence Maximum length, dictionary size of token\_type\_ids, stdev of truncated\_normal\_initializer.

Followed by Word Embedding Construct embedding\_table, perform word embedding, optional one\_hot, return embedding result and embedding\_table.

Then the attention mask is constructed. Convert the 2D mask whose shape is [batch\_size, to\_seq\_length] into a 3D mask whose shape is [batch\_size, from\_seq\_length, to\_seq\_length] for attention.

Finally, transformer. The input from\_tensor is used as query, and to\_tensor is used as key and value. When self attention, from\_tensor and to\_tensor are the same value. At the beginning, the function checks the input shape and obtains batch\_size, from\_seq\_length, to\_seq\_length. If the input is a 3D tensor, it will be converted into a 2D matrix (take word\_embedding as an example [batch\_size, seq\_lenth, hidden\_size] -> [batch\_size\*seq\_lenth, hidden\_size]). The query\_layer, key\_layer, and value\_layer are generated through fully connected linear projection, and the second dimension of the output becomes num\_attention\_heads \* size\_per\_head (the entire model defaults to hidden\_size=num\_attention\_heads \* size\_per\_head). Then convert to long by transpose\_for\_scores. Calculate attention\_probs (attention score) according to the formula. If attention\_mask is not None, add a large negative number to the mask part, so that the corresponding

probability value after softmax is close to 0, and then dropout. Finally, multiply value and attention\_probs to return a 3D tensor or 2D matrix.

### 3.3 SCIBERT

SCIBERT [1] uses unsupervised pre-training in large multi-field science publications to improve the performance of downstream science NLP missions. SCIBERT follows the same architecture as BERT, but is pre-trained in scientific texts.

BERT USES WordPiece for unsupervised tagging of input text. Build the vocabulary to include the most commonly used words or subword units. SCIBERT referred to BERT's original vocabulary as base-vocab.

The model USES the SentencePiece library to build SCI-Vocab, a new WordPiece vocabulary from our science corpus. SCIBERT generates case-sensitive and case-sensitive vocabularies and sets the vocabulary size to 30K to match the base-Vocab size. The resulting token overlap between base-vocab and SCI-Vocab was 42%, indicating a significant difference in common terms between the text of the scientific field and that of the general field.

SCIBERT was trained in 1.14 million random samples from Semantic Scholars. The corpus consists of 18 per cent of papers in computer science and 82 per cent of papers in the broad biomedical field. The model USES the full text of the paper, not just an abstract. The average paper length was 154 sentences (2,769 tokens), resulting in a corpus size of 3.17b tokens, similar to training BERT's 3.3b tokens. It splits sentences using a ScispaCy optimized for scientific text.

BERT training for long sentences can be slow. Following the original BERT code, SCIBERT sets the maximum sentence length for 128 tokens and trains the model until the training losses stop decreasing. The model then continues to train the model to allow up to 512 token sentence lengths.

SCIBERT use Bert-base pretrain weights. From the result, SCIBERT significantly outperformed BERT-Base and achieves new SOTA results on several of these tasks, even compared to some reported BIOBERT results on biomedical tasks.

### 3.4 COVID-19 Model based on SCIBERT

Compared with the BERT model, SCIBERT has a more professional understanding of scientific and medical literature, and can more effectively complete downstream NLP tasks for scientific and medical text. On this basis, we guessed whether more specialized and trained models can have a more prominent performance in the specialized literature. Based on this conjecture, we designed a SCIBERT model that uses the COVID-19 literature collection for re-training. The model is trained on 50,000 abstracts of COVID-19 papers from CORD-19, and the average sentence length is 120 words.

We have collected a corpus on COVID-19 from the website. In order to facilitate training, the collection of papers needs to be integrated. First, we wrote a script to extract the abstract of each document. Here, due to the limitation of computing power, we only extract the abstract part of the article as input instead of the article body, which is different from the pretrain process of the SCIBERT model. In the extracted file, each line contains a sentence, and each document is separated by a blank line. This is to enable the NSP training process. The literature collection also contains some articles in other languages. We have removed non-English literature to ensure that training is not disturbed by outliers.

Our model structure uses the same model structure as SCIBERT. We use one single GPU with 4 cores for training. The SCIBERT-C19-1.0 model is pre-trained based on SCIBERT. The training document set consists of 10,000 articles about COVID-19. Here we only extract the abstract part of each article as the training input. The maximum sequence length is 128, the number of training steps is 10000, the times of warmup training is 1000, and the learning rate is  $2e-5$ . In the warmup phase, the learning rate will be lower than after, and then the learning rate will return to normal. We hope to observe whether the model has changed through small-scale training first. It took about 2 hours to train this model.

The advanced SCIBERT-C19-2.0 model is also based on SCIBERT for training. The training sample size is increased to 50,000 articles about COVID-19, the number of training steps is increased to 20,000, the times of warmups is 100, and the learning rate is  $2e-5$ . This model takes about 5 hours to train.

Our task is to collect features, so no further fine-tuning is performed.

### 3.5 Clustering

Due to the lack of literature labels in the literature data set, it means that this will be an unlabeled unsupervised learning. We use a clustering algorithm for simple classification.

Clustering is the division of a data set into different classes or clusters according to certain criteria (commonly use distance criteria), so the goal is to expect data objects in the same cluster to be as similar as possible, and data objects in different clusters to be as different as possible. In other words, after clustering, try to gather the same type of data together and separate different data as much as possible. Clustering technology is used in many fields, such as image processing and user portrait has a wide range of applications.

For texts, we hope that the more similar texts are gathered together as much as possible, and then through other precautions, the understandable classification meaning of each cluster can be further summarized.

Typical clustering methods include:

#### 3.5.1 K-Means

K-Means clustering algorithm is an iterative clustering analysis algorithm. The main idea is to first divide the data into K groups, then randomly select K objects as the initial clustering center point, calculate the distance between each object and each seed clustering center point, and assign each object to the clustering center nearest to it. This ensures that each object belongs to the same class as its nearest central point, thus forming a cluster around the cluster center and the objects assigned to them. This construction process is repeated until a termination condition is met. The specific algorithm is introduced in many articles and many improved algorithms are derived.

When applying the K-Means method, the most important thing is to determine the number of clusters and the distance calculation method. The method to determine the K value is the Elbow diagram method. By trying different K values, and drawing the loss function corresponding to different K values into a line graph, the horizontal axis is the value of K, and the vertical axis is the loss defined by the sum of squared errors. Function, the Elbow diagram method considers the inflection point to be the best value of K. Elbow diagram is an empirical method. The disadvantage



is that it is not automated enough. Therefore, the research institutes have proposed some more advanced methods, including the more famous Gap Statistic method.

The advantage of the Gap Statistic method is that it does not require manual judgment, but only needs to find the  $K$  corresponding to the largest Gap Statistic, so this method is suitable for batch operations. Here we use the same loss function. When the cluster is  $K$  clusters, the corresponding loss function is denoted as  $D_k$ . Gap Statistic is defined as:

$$\text{Gap}(K) = E(\log D_k) - \log D_k$$

Where  $E(\log D_k)$  is the expectation of  $\log D_k$ , which is generally generated by Monte Carlo simulation. We randomly generate as many random samples as the original samples according to the uniform distribution in the area where the samples are located, and do K-means for the random samples, and get a  $D_k$  repeated many times to calculate the approximate value of  $E(\log D_k)$ . The actual meaning of  $\text{Gap}(K)$  is the difference between the loss of a random sample and the loss of an actual sample. When  $K$  is the optimal number of clusters corresponding to the sample, then the loss of the actual sample should be relatively small, and the difference between the random sample loss and the actual sample loss should also be minimized, so that the  $K$  value corresponding to the maximum value of  $\text{Gap}(K)$  is the best number of clusters.

At the same time, this method is also susceptible to the selection of random initial center points. If the initial cluster center is not well selected, it is easy to fall into a local optimal solution. In response to this problem, the researchers proposed an improved method K-Means++, which improved the selection of the initial center point of the K-Means algorithm. A simpler method is to run K-Means multiple times, and finally take an average result.

This article uses the K-Means method in the scikit-learn toolkit. The K-means algorithm aims to choose centroids that minimise the inertia, or within-cluster sum-of-squares criterion. The K-Means method can also use K-Means++ to select the initial center point during initialization, and the parameter `init` needs to be set.

### 3.5.2 DBSCAN

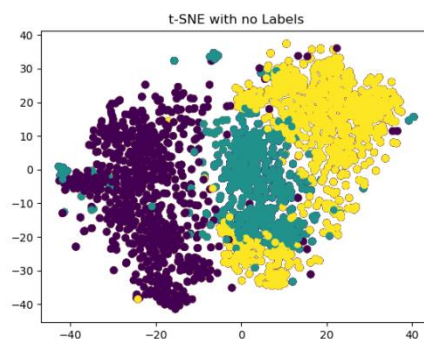
DBSCAN is a clustering method based on connectivity and density functions. In using DBSCAN for clustering, we found that it creates dimensional disasters when applied to high-dimensional data clustering. At the same time, computational complexity increases dramatically as the dimension increases. Therefore, it is not suitable for this project, and it is also proved in the following demonstration.

### 3.5.3 BIRCH

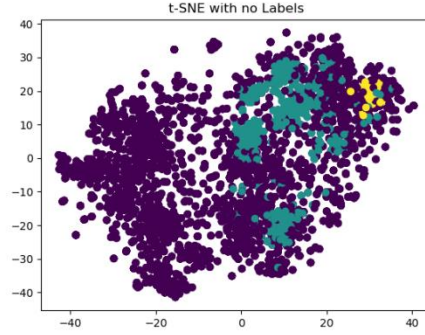
BIRCH's full name is Facsimile Reduction and Clustering using Hierarchies, a method based on the hierarchical decomposition of data sets (or objects).

BIRCH Algorithm uses tree hierarchies to achieve fast clustering. This classification structure is similar to a balanced B+ tree. It is often referred to as the clustering feature tree (CF tree for short). Each node of the tree consists of several clustering characteristics. Each node containing leaf nodes has several clustering characteristics. The clustering characteristics of internal nodes have Pointers to child nodes. All leaf nodes are linked by a bidirectional linked list.

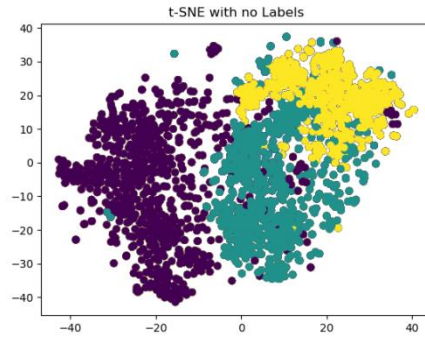
After sorting out three different types of clustering methods, we hope to select a clustering method suitable for our text clustering task. Based on the clustering results, a preliminary effect can be generated first. Taking the results of SCIBERT-C19-1.0 model clustering 3000 documents as an example, the results of three clustering methods are visualized as follows:



**Figure 14. K-Means**



**Figure 15. DBSCAN**



**Figure 16. BIRCH**

It can be seen that the K-Means model and the BIRCH model can better complete the clustering, and the density-based DBSCAN algorithm is insufficient in distinguishing each category, so we do not choose DBSCAN for clustering.

### 3.6 Evaluation

After using the clustering method on the data set, the quality of the resulting clusters needs to be evaluated. The clustering algorithm is an unsupervised method, and the common evaluation content includes the degree of clustering within a class and the degree of dispersion between classes. As an unsupervised learning task, it is very necessary to evaluate the effect of clustering, otherwise the results of clustering will be difficult to apply.

The evaluation methods of clustering are divided into two categories in general, one is to analyze external information, and the other is to analyze internal information. The external

information is the visual information that can be seen directly, and it refers to the cluster number after the clustering ends. However, this method cannot be applied, because it is necessary to artificially judge whether the clustering result is correct or not, which is impractical when the amount of data is large. Another way to analyze internal information is that the clustering results will generate parameters such as entropy and purity through some models. Without using external information for evaluation, internal information evaluation does not require manual identification of the clustering quality, which means that you do not need to know what the clustering label is, only the evaluation index. And in many cases, the labels after clustering are invisible, so that we can only use internal information to evaluate the quality of the clustering.

Currently, there are many proposed clustering algorithm evaluation algorithms, such as CH, DB, SD, S\_Dbw, etc. However, these algorithms may be affected by data anomalies, such as noise in the data. Consider five types of data anomalies that may predict accuracy: monotonicity, noise, density, subcluster, and skewed dispersion. In each case, these evaluation indicators will be used to test, and finally found that the S\_Dbw algorithm is more stable.

### 3.6.1 S\_Dbw

We introduce a new evaluation index S\_Dbw to evaluate the clustering results. S\_Dbw approach performed favorably in all cases, even in those that other indices failed to indicate the correct partitions in a data set. The article [3] proves the reliability and value S\_Dbw methods empirically and theoretically.

S\_Dbw is composed of two items, namely the separation between clusters and the compactness within clusters. When using it to evaluate the results of the clustering algorithm under different parameters, select the set of parameters with the smallest S\_Dbw value.

$$stdev = \frac{1}{c} \sqrt{\sum_{i=1}^c \|\sigma(v_i)\|}$$

$$density(u) = \sum_{l=1}^{n_{ij}} f(x_l, u) \begin{cases} 0 & \text{if } d(x, u) > stdev \\ 1 & \text{otherwise} \end{cases}$$

$$Dens\_bw(c) = \frac{1}{c \times (c - 1)} \sum_{i=1}^n \left( \sum_{\substack{j=1 \\ i \neq j}}^c \frac{density(u_{ij})}{\max(density(v_i), density(v_j))} \right)$$

$$Scat(c) = \frac{1}{c} \sum_{i=1}^c \frac{\|\delta(v_i)\|}{\|\delta(S)\|}$$

$$S\_Dbw(c) = Dens\_bw(c) + Scat(c)$$

The experiment shows that the result of the index does not depend on the clustering algorithm used, that is, the performance is stable under different clustering algorithms, and the optimal input parameter of the algorithm is always pointed out in each case. The experimental evaluation results show that the effectiveness index of this method is better than that of the latest one in the literature.

### 3.6.2 Silhouette Coefficient

Silhouette Coefficient is often used to compare different quantitative clustering results from the same algorithm. Here we introduce it as a reference index to compare the performance of the model under different clustering parameters. Silhouette Coefficient will range from -1 to 1. When the Silhouette Coefficient value of the data point P is close to 1, the clustering containing P is relatively compact, and P is far from other clustering, which is an ideal situation. Note that a negative value of Silhouette Coefficient means that P is closer to an object from the other cluster than from the same cluster. This situation is very bad in clustering tasks and should be avoided.

### 3.6.3 Calinski-Harabaz Index

The Calinski-Harabaz Index can better reflect the differences in the clustering effects of different models, so for the clustering results, we add Calinski-Harabaz Index for evaluation.

The index equals to the ratio of two parts. One is the sum of between-clusters dispersion and the other is the inter-cluster dispersion for all clusters. Dispersion is defined as the sum of distances squared. The formula is as follow:

$$s = \frac{tr(B_k)}{tr(W_k)} \times \frac{n_E - k}{k - 1}$$

Where  $B_k$  and  $W_k$  are the two dispersion matrix:

$$W_k = \sum_{q=1}^k \sum_{x \in C_q} (x - c_q)(x - c_q)^T$$

$$B_k = \sum_{q=1}^k n_q (c_q - c_E)(c_q - c_E)^T$$

Compared with the Silhouette Coefficient, the advantage of the Calinski-Harabaz Index is that the calculation is faster, with a difference of several hundred times and milliseconds.

### 3.7 Discussion

We discuss the clustering results of each model. In the clustering process, except for the different models, the settings of other variables are the same (batch\_size, max\_seq\_length, etc). First, we use the original BERT model for feature extraction. Second, we use the SCIBERT model for feature extraction. Finally, based on the SCIBERT model, we added COVID-19-related literature collections for pre-training, and used the retrained model SCIBERT-C19 for feature extraction.

### 3.8 Quantitative Analysis

We used a collection of documents containing 3000 articles on COVID-19, used different models to perform cluster analysis, and calculated the results of cluster evaluation data as follows. In the table below, we compare the trained SCIBERT-C19 model with the BERT and SCIBERT models.

**Table 2.** Table of Clustering Performance

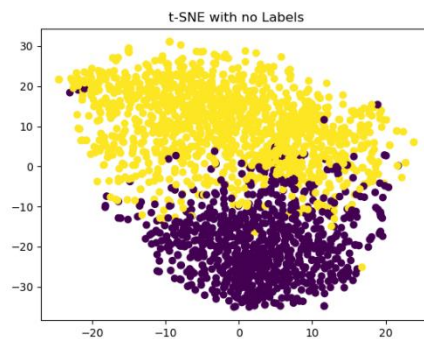
#clusters	index	BERT	SCIBERT	SCIBERT-C19-1.0	SCIBERT-C19-2.0
2	CH Index	221.7686	140.2639	155.710429	<b>160.91585</b>
2	Silhouette	0.099174	0.066726	<b>0.0901453</b>	0.08990748
2	S_Dbw	0.944034	<b>0.96569</b>	0.98914766	0.98510225
3	CH Index	/	107.1546	108.572770	<b>112.895236</b>
3	Silhouette	/	<b>0.05581</b>	0.03624489	0.03496972
3	S_Dbw	/	0.954935	0.93868355	<b>0.92872932</b>
4	CH Index	/	91.08114	90.8312951	<b>94.468591</b>
4	Silhouette	/	<b>0.04337</b>	0.03664264	0.042989720
4	S_Dbw	/	0.941559	<b>0.9410767</b>	0.959846483

The data sets used for clustering are all articles related to COVID-19, which themselves have a strong degree of similarity because the common themes. The BERT model with strong generalization ability can only generate clustering results when the number of clusters is two due to the tight distribution of clustering results. Although the quantitative indicators look better, there are still shortcomings in further analysis (see Visual Analysis for details).

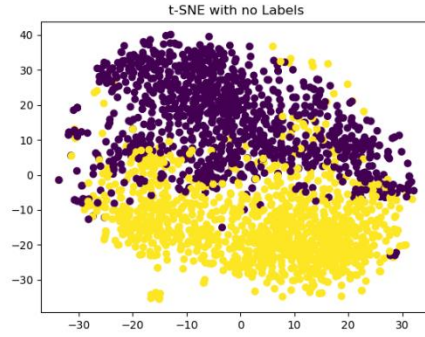
In SCIBERT and its derivative models, it can be found that as the training level increases (From 10,000 to 50,000), the clustering results have improved. Specifically, the Calinski-Harabaz Index has improved significantly under all conditions of different number of clusters. When the number of clusters is equal to 2, the improvement effect is most obvious (+20.7). When the number of clusters is equal to 2, the Silhouette Index is significantly improved, and it is higher than other numbers of clusters. This shows that the number of clusters equal to 2 is the clearest clustering result, which can also be intuitive in visual analysis. see. The S\_Dbw index does not change much in different models, which verifies the stability of each model.

### 3.9 Visual Analysis

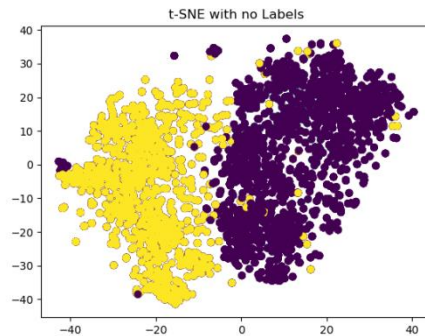
The dimension of the text feature vector extracted by the BERT, SCIBERT and SCIBERT-C19 models is 768 dimensions, which cannot be directly visualized. We use the t-SNE algorithm to visualize the dimensionality reduction of high-dimensional text feature vectors, and select the clustering results of the K-Means algorithm with the number of clusters equal to 2 for visual demonstration. The results are as follows:



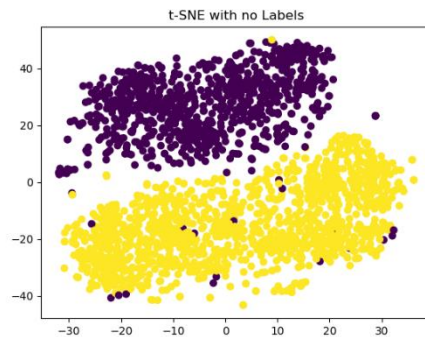
**Figure 17. BERT**



**Figure 18. SCIBERT**



**Figure 19. SCIBERT-C19-1.0**



**Figure 20. SCIBERT-C19-2.0**

It can be seen from the results that the original BERT model has a high degree of feature vector aggregation---pay attention to its coordinate axis----so it can be judged that the BERT model cannot deeply distinguish the internal differences of the COVID-19 category of documents. The SCIBERT clustering result is significantly improved compared to BERT from the image. Its



point distribution is more scattered, which means that it can more deeply distinguish the internal differences of COVID-19 type documents.

SCIBERT-C19 series models have better performance on the basis of SCIBERT. In the 1.0 model obtained by training with a small number of samples, separation between clusters and clustering within clusters are produced. As the number of training samples increases, the interval between clusters in the 2.0 model is more obvious. As shown in Figure 4, there has been a clear separation between the two parts. Therefore, it can be proved that the SCIBERT model based on COVID-19 training has better performance in dealing with COVID-19 related literature.

### **3.10 Text Similarity Calculation**

Text matching algorithms are mainly used in search engines, question and answer systems, etc., to find the most relevant text to the target text. For example, information retrieval can be reduced to the matching of query items and documents, the question and answer system can be reduced to the matching of questions and candidate answers, and the dialogue system can be reduced to the matching of dialogue and replies.

The traditional text matching model is based on literal matching. By comparing whether each character in the two strings is equal, you can know whether the two strings are equal, or it is simpler to map each string to a hash value through a hash function. , And then compare. Specific methods include TF-IDF, BM25, simhash and so on.

What we expect is semantic-based text matching. The text semantic feature matrix is obtained through the BERT model for text similarity calculation, and the calculation method is generally through cosine similarity.

Commonly used methods for judging vector similarity include Euclidean distance and cosine distance, each of which has different calculation methods and measurement characteristics, so they are suitable for different data analysis models.

Euclidean distance can reflect the absolute difference of numerical characteristics of objects, so it is mainly used to analyze problems that need to reflect the difference from the numerical scale,

such as analyzing the similarity or difference of user characteristics by using user behavior indicators.

The cosine distance focuses on distinguishing between directional differences and is therefore insensitive to absolute values. It is more used to distinguish the similarity and difference of user characteristics by using users' scores on multiple contents, and also solves the problem of possible non-uniform measurement standards between users, because cosine distance is not sensitive to absolute value.

Most of the Natural Language Processing tasks have proved that the benchmark average word embedded cosine similarity lookup method has good performance. Cosine similarity ranges from minus 1 to 1. If you look at the graph, the bigger the cosine of the angle, the smaller the angle between the two vectors, the bigger the angle between the two vectors. When the two vectors coincide in the direction, the cosine of the angle between them is at the maximum value of 1, which means they are completely similar; when the two vectors are in the opposite direction, the cosine of the angle between them is at the minimum value of -1, which means they are completely different.

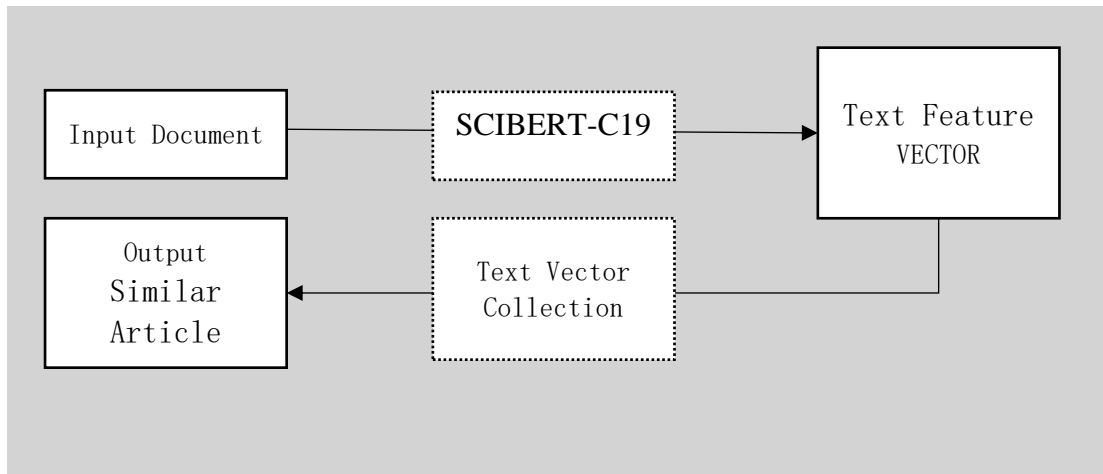
### 3.11 Similarity matching

For the extracted text feature vectors, similarity matching can be used to find the text with the highest similarity and realize the function of similar text search. We use cosine similarity to calculate text similarity, the formula is as follow:

$$\cos(\theta) = \frac{\sum_{i=1}^n (x_i y_i)}{\sqrt{\sum_{i=1}^n (x_i)^2} \times \sqrt{\sum_{i=1}^n (y_i)^2}} = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \times \|\mathbf{b}\|}$$

Where a and b are the vectors we need to compare.

In order to speed up the processing speed of the website and reduce the calculation steps of each similarity matching, we used the model to extract 3000 documents in advance-the number of documents can be more-integrated the text feature vectors of all documents into one document. In the actual application process, only the feature vector of the input text needs to be extracted to calculate the cosine similarity, without re-extracting the feature of the document collection, which improves the efficiency of the query and reduces the waiting time of users.



**Figure 21.** Text matching process

### 3.12 Conclusion and Future Work

The advent of the BERT model is a major breakthrough in the NLP field and a milestone in language representation. At present, a large number of models developed based on BERT and applied to different fields have come out one after another. SCIBERT is a very typical specialty BERT model. We stand on the shoulders of giants and attempt to conduct more in-depth research and application in the scientific field, so we try to build a language model that specializes in COVID-19. On the other hand, COVID-19 is still raging around the world, and we hope to make some contributions to this.

This paper analyzes the main problems faced by text feature extraction, and tries to use new generative and extractive feature extraction as the goal, and starts the work. We introduced the current best natural language processing task pre-training model BERT, and combined with the more professional SCIBERT model in the scientific field, using BERT sentence vectors to replace traditional word vectors to obtain more deep semantic features, and proposed SCIBERT - COVID19 specialized model.

The results of the model roughly meet our expectations, and there has been a significant improvement in the quantitative text feature recognition indicators. Our ultimate goal is to instantiate the model. We believe that models with better recognition capabilities can provide users with more accurate results.

However, our model still has some shortcomings and needs to be improved. Due to the limitations of the data set and computing power, what we can currently obtain is an initial model, which contains fewer training sets and fewer training times. However, from the perspective of the feature extraction effect, SCIBERT-C19 has been significantly improved, and it has a better performance in reading the COVID-19 literature.

In the selection of the similarity matching method, we simply chose the cosine similarity, but for the processing of high-dimensional data, some key information may be lost. This is worthy of further research in the future.

In future work, we hope to continuously expand the training set database, let the existing SCIBERT-C19 model perform reinforcement learning, and make it an expert on COVID-19.

## **4. Website building**

To allow the scientific researchers to use our research results more conveniently, we have established an article clustering search website with a friendly interface, rich resources, and accurate results after user demand analysis. Users can easily use our model to look for similar articles quickly. The website's front-end uses CSS, HTML, and other website interface design languages, JavaScript programming language, and Vue.js website design framework. The back-end uses Python language, including Flask and other network programming libraries and TensorFlow and other scientific training libraries, and the network interface uses AXIOS and Nginx, the database uses the most convenient MySQL and the entire website realizes the functions of users uploading files, searching articles, and finding similar documents.

This website aims to respond to the current situation of the explosive growth of newly published coronavirus literature caused by the rapid spread of the epidemic and to provide scientific researchers around the world with a high-quality article clustering and similar document search engine, thereby helping researchers to understand the current research status of the coronavirus.

### **4.1 Research status**

With the rapid rise of network information technology, related article search functions are trendy. However, at this stage, there is still no article clustering website dedicated to the new

coronavirus, and there is no related similar recommendation system based on natural language processing. With the outbreak of the COVID-19 in the world, this demand has become more and more urgent. The publication of thousands of articles every day makes it difficult for frontline researchers to obtain the latest research progress, which greatly slows down the research progress.

At the same time, most of the document recommendation algorithms at this stage are not based on full text but use title matching for the recommendation or through a collaborative recommendation system for the recommendation. This will give more weight to papers that have already been highly popular, but papers that have not received attention at the beginning cannot be well recommended. Therefore, such a recommendation system greatly prevents the buried papers from showing their true value, especially in the current information explosion stage. Therefore, we need to analyze all the papers indiscriminately and train through natural language processing so that we can play the value of each paper to a greater extent and make those valuable papers discoverable by researchers. Therefore, we need a brand new paper recommendation idea and matching models and algorithms to assist us in implementing our recommendation system.

#### **4.2 The goal of this website**

This website's main research content is to develop a document clustering and document similar recommendation websites based on our own trained BERT model and LDA model and establish our own database to more easily store and read data. It also provides scientific researchers with a fast, convenient, and accurate new coronavirus literature search comprehensive website, which is of great significance for frontline researchers to obtain similar literature quickly. Users only need to click on our homepage to operate quickly. The main functions of our website are as follows:

1. Users can upload the documents they downloaded (JSON format or PDF format) to get the recommendation of similar documents and the introduction of related clusters;
2. Users can also quickly get the recommendation of similar documents and the introduction of related clusters by entering the DOI number of the document;
3. Users can quickly understand our model and our results by introducing our project model on our website so that we can use our model more flexibly.

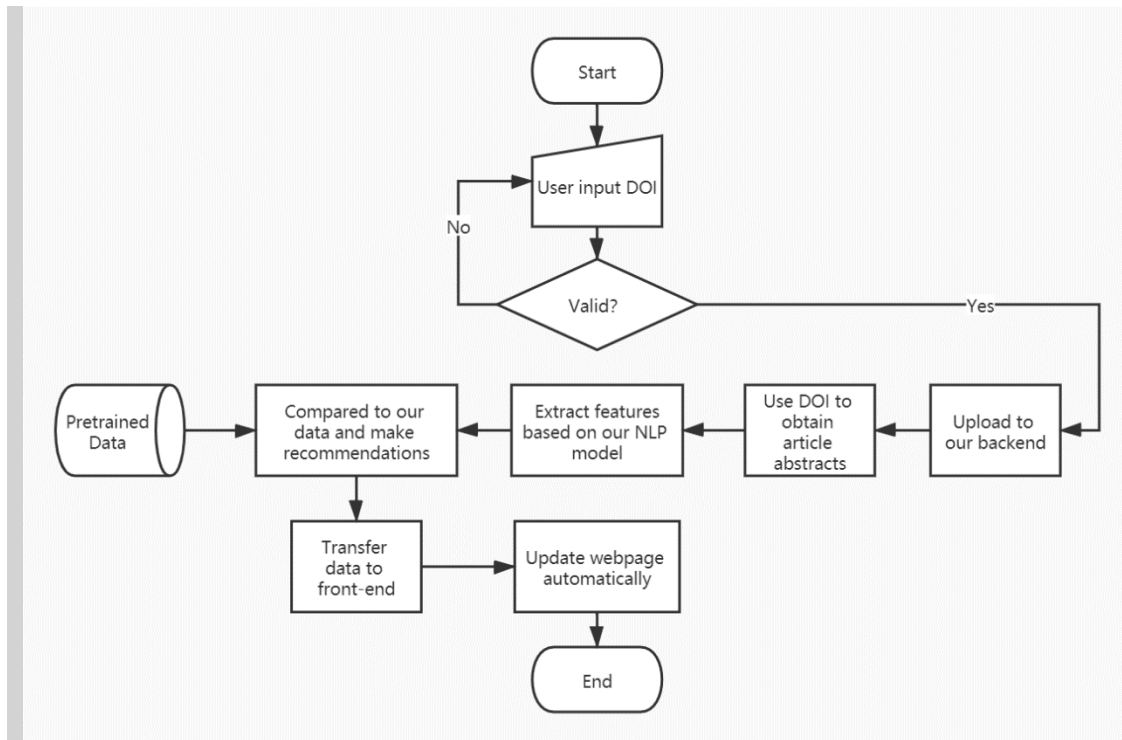
### **4.3 Design Principles**

In order to ensure the smooth design and development of the website system and the successful realization of each expected function, the following principles must be followed:

1. **Safety and reliability:** In developing the system, it is necessary to ensure that the input and output data of the system have a certain degree of safety and reliability to ensure the accuracy of the system. The host, database, software platform, and browser platform used must ensure the system's security and reliability.
2. **Simplicity and consistency:** The layout style of each page of the website should be unified, and the color tone should be coordinated and clear. Then visual optimization should be carried out according to different pages' characteristics to achieve good aesthetics and clear layering. It can make users better understand and operate.
3. **Practicality and economy:** The websites are based on the original intention of providing convenience for the vast number of scientific researchers. The development platform and resources used in the development and design process are completely free and open and have certain practicability and economy.
4. **Scalability:** The development platform used in the system development and design process has good scalability. With the continued development in the future, the system is still compatible and easy to implement.

### **4.4 System functional structure overview**

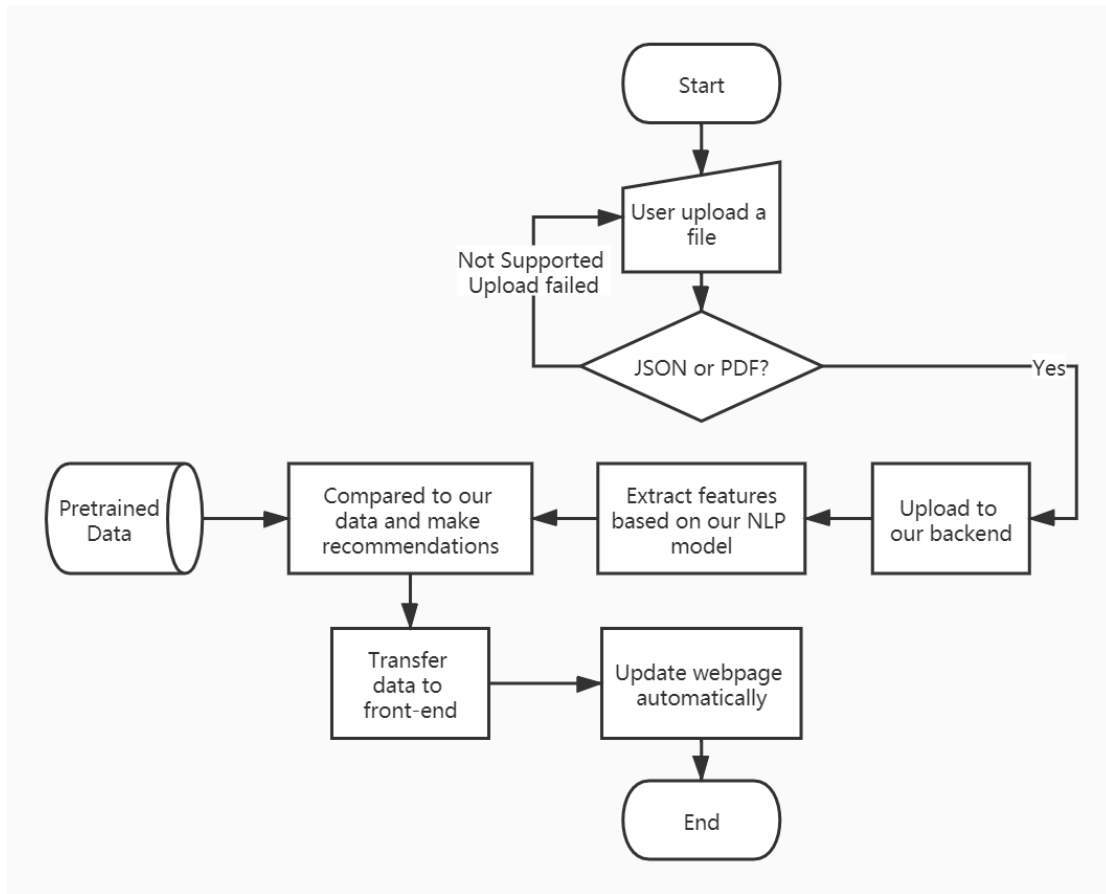
There are two ways for the users to get the content of the documents they are interested in. Users can input DOI or upload PDF or JSON format file. Through these two methods, users can quickly obtain article clustering and similar article recommendation functions. The system flowchart of the two functions is shown below.



**Figure 22.** The First Method's Flow Chart of Our Website

First of all, see Figure 22, which is the first mode of operation of our website. We ask users to enter the DOI number of the article they are interested in. If the user does not know the DOI number, we also attach relevant hits on the website. When the user enters the DOI number and clicks the search button, the DOI number will be automatically transmitted to our back-end. When our back-end judges that the DOI number is correct, we will use this DOI number to obtain the document's relevant content. We use the web crawler method here. We use this DOI number to build a Baidu academic query website. We use the Selenium package in Python to crawl. After loading this website in the background, if the DOI number is correct, the page will automatically jump to the related literature's details page. We collect some relevant information by locating the elements in the details page, including the title of the document, author, publication time, abstract, etc. However, due to the limitation of the literature's copyright, we cannot use this crawler to crawl the article's text, for the time being, so we only use the article's abstract for the time being. After obtaining the relevant information of the article, we will use our own trained BERT model to extract the features in the article and compare it with the feature values of the extracted features in our database, and use the cosine similarity algorithm to calculate the similarity degree, and then get our similar literature. We connect to our database to obtain information about similar documents,

including title, author information, links, clustering information, clustering keywords, and other details, and return these results to our front-end. Our front-end will automatically update our form after receiving the information from the back-end, and users can get our recommendations by viewing the form below us.



**Figure 23.** The Second Method's Flow Chart of Our Website

Next, see Figure 23, which is the second mode of operation of our website. We require users to upload the corresponding files. The file formats we support include PDF format or JSON format. The JSON format is equal to the JSON format of our data source mentioned in the previous article. This upload method can ensure that all the content in our article can be calculated. Next is the PDF format. We will use the library of PDF format decoded in Python for analysis. At present, we only select the content of the first page for calculation, so we will require that the first page be the first page of the document instead of the database's introduction page. After we parse the PDF



information, since we cannot distinguish the title, author, abstract, and body in a wide variety of formats, we will pass all the content of the first page to our model for calculation, which may cause some deviation. So our web pages also strongly recommend our users to use DOI input for inquiries. After the user uploads the file, after we parse the information we need, we will use our own trained BERT model to extract the features in the article and compare it with the feature values of the extracted features in our database. Use the algorithm of cosine similarity to calculate the degree of similarity, and then get our similar documents. We connect to our database to obtain information about similar documents, including title, author information, links, clustering information, clustering keywords, and other details, and return these results to our front-end. Our front-end will automatically update our form after receiving the information from the back-end, and users can get our recommendations by viewing the form below us.

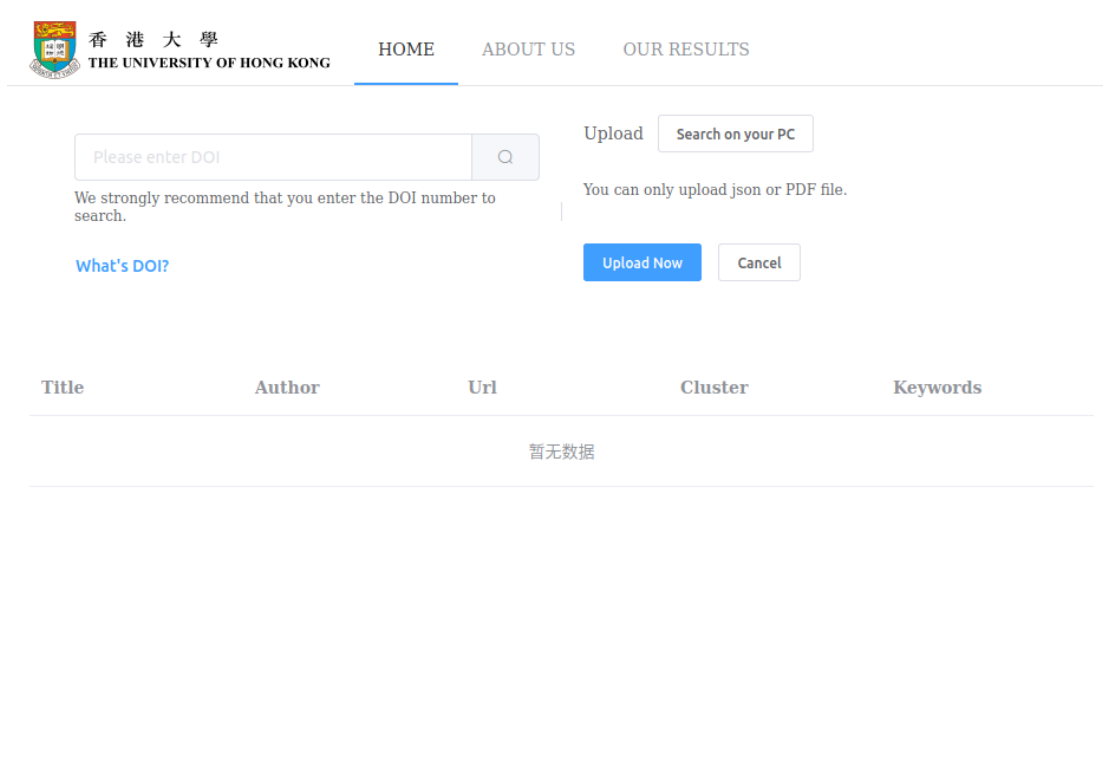
We will not repeat our specific code details here because of the large amount of code. If you are interested in our code, please contact us, and we will provide you with some of our key codes for communication and exchange.

#### **4.5 System core technology analysis**

This website system's core technology can be divided into two parts: front-end and back-end development and server construction.

1. The front end uses CSS, HTML, and other website interface design languages, JavaScript programming language, and Vue.js framework to build an adaptive website interface, to realize the good use of the website on multiple devices and complete the front-end design of the website and some front-end and back-end interactions.
2. The back-end uses Python language, integrates network programming libraries such as Flask and scientific training libraries such as TensorFlow so that it can make good use of our trained models. A back-end program with complete functions was initially built.
3. The deployment of Nginx on the cloud server can already provide external access services. Deploy front-end static resources in Nginx, which supports ultra-high concurrency, and use Nginx's user ports to access multiple applications to achieve reverse proxy.

## 4.6 Web page operation



The screenshot shows the homepage of a website associated with The University of Hong Kong. The header includes the university's name in Chinese and English, and navigation links for HOME, ABOUT US, and OUR RESULTS. The main content area features a search bar with the placeholder text 'Please enter DOI' and a search button. Below the search bar, a message states: 'We strongly recommend that you enter the DOI number to search.' A link 'What's DOI?' is provided. To the right, there is an 'Upload' section with a 'Search on your PC' button. Below this, a message says: 'You can only upload json or PDF file.' There are two buttons: 'Upload Now' and 'Cancel'. At the bottom, there is a table with the following columns: Title, Author, Url, Cluster, and Keywords. The table currently displays '暂无数据' (No data).

**Figure 24.** The screenshot of our website

Users only need to enter our website, and the home page is the main function we can provide. The user only needs to click the input box to enter the DOI number and click the search button, and our background will automatically run our model to get our recommendation results. Users can also click the blue *What's DOI?* to gain basic information about DOI.

Users can also click the *Search on your PC* button to open the dialog box, select the document (PDF format or JSON format) you want to upload in the dialog box, and click the *Upload Now* button to transfer the file to our servers. We will also automatically analyze the article and give users our clustering results and recommendation results.

The following table will be automatically updated after users upload documents or enter DOI and provide the titles, authors, links, clusters, keywords of similar documents. The user can quickly enter the document's detail page by clicking the URL and then quickly download or obtain related information. Cluster is the result of clustering by our model. Keywords are the keywords of the cluster. The specific details can be obtained by clicking the *OUR RESULTS* tab.

Users can click our results to quickly get our model results, including LDA clustering results and keyword information, as well as the results of our own trained BERT model.

It can be seen that the overall process of our website is very smooth, and the user can quickly find the information he needs without unnecessary operations. At the same time, we also need to make some small tips here. First of all, the way we get the most accurate results is to upload a JSON file in the correct format, but this may waste a lot of time, and we may make improvements to this point in our future work. So our most recommended way is to enter the DOI number to make a query so that our speed is not only fast, but our effect will be relatively more accurate. Secondly, after we check the relevant literature, we can quickly click on the link we provide to link to the literature's detailed page. We recommend that you use the database of your university or related scientific research institution to query this literature to obtain the download address. Due to copyright restrictions, we are temporarily unable to provide users with direct link downloads.

#### **4.7 Conclusion and Discussion**

We successfully built the front-end using CSS, HTML and other website interface design languages, JavaScript programming language and Vue.js website design framework, Python language, and its Flask and other network programming libraries and TensorFlow and other scientific training libraries to build the back-end, using AXIOS and Nginx and other services have built network services, and built our database with free and easy-to-use MySQL, and successfully built a friendly and easy-to-use website. The majority of scientific researchers can use our research results more conveniently, enabling users to upload files, search for documents, and find similar documents. At the same time, it ensures timeliness, abundant resources, and accurate results. It can effectively respond to the current situation of the explosive growth of new coronavirus literature caused by the rapid spread of the epidemic, and provide scientific researchers around the world with a high-quality document clustering and similar document search function.

However, our website still needs to be improved. For example, it has not passed some security operation and maintenance tests, including stress resistance tests, anti-attack tests, etc. Due to the current server problems, the speed is not very fast. Therefore, we will use more advanced technology to protect our website in the future to serve the vast number of scientific researchers better. At the same time, our website is also considering some new features, including more

targeted queries in different clusters, and users can upload data independently to enrich our database. These functions will be developed after a careful needs assessment. At the same time, we also plan to add some additional features to facilitate users to obtain more accurate results. First of all, we plan to add a form search box. Users can copy the relevant document content and paste it into the corresponding input box (such as title, author, abstract, body, etc.) so as to make more accurate queries more convenient. Simultaneously, we will also communicate with various databases and may directly provide links to related documents for more convenient downloading.

## **5. Results and findings**

The combination of BERT and K-Means realizes the extraction of text features, vectorization representation, and clustering. The approach divides a large collection of articles into several categories according to their abstract semantics and labels them. Based on abstract semantic features, cosine similarity realizes the similarity matching function using the feature vector of the abstracts. The model we trained roughly meets our expectations, and there has been a significant improvement in the quantitative text feature recognition indicators. From the perspective of the feature extraction effect, SCIBERT-C19 has been significantly improved, and it has a better performance in reading the COVID-19 literature.

We successfully used LDA and other models to extract keywords, selected many different solutions for comparison, and selected the best solution. The evaluation results show that our last two programs will be more effective. The previous link also mentioned that the reason for guessing is that the LDA variable and the BERT variable use autoencoders to reduce the size instead of mixing them into the same autoencoder for training as in the previous method. The LDA variable is the summary of the entire article, and the Bert variable represents each word, which obtains the local characteristics of the article. Therefore, it is best to merge them in the final stage, rather than continue to use other methods after the merger. Through these methods, we can effectively extract the keywords in the cluster, which can help researchers quickly grasp the cluster's key meaning to better understand which cluster or clusters their research direction is in. In this way, you can make more targeted queries. These models have greatly increased scientific researchers' efficiency to a certain extent, saved the time of scientific researchers, and made certain contributions to the fight against the epidemic.

We used these models to build a friendly, convenient, and free website, which can provide researchers with a system for document clustering and document similarity recommendation. Through this website, the researcher can easily obtain the clustering results and similar recommendations of the documents he is interested in. By simply entering DOI or uploading documents in JSON or PDF format, our system will run automatically, using our model to provide researchers with our calculation results within a few seconds, including the titles, authors, links, and clustering results and clustering keywords, etc. of similar documents. Researchers can use this information to find similar documents, thereby improving scientific research efficiency quickly. Moreover, researchers can easily view and use our model results on our website.

## **6. Future plans**

Our trained Bert model's results roughly meet our expectations, and there has been a significant improvement in the quantitative text feature recognition indicators. However, our model still has some shortcomings and needs to be improved. First, we need to expand our current training set. Furthermore, call more powerful computing resources for training in order to improve the effectiveness of our model. Meanwhile, our ultimate goal is to instantiate our model so that our model can provide better results for users and play more value for others. Secondly, we are working on the similarity matching algorithm. We only consider one algorithm, for the time being, that is, the cosine similarity algorithm. This algorithm may lose key information when processing high-latitude information. We hope that we may find a better similarity matching algorithm in future work to improve our calculation results' accuracy further.

Regarding the future work of the LDA model and the BERT model, we still think that there are many areas for improvement. As mentioned earlier, we can improve our LDA model, such as focusing on the order of words, using label expansion models, etc. Moreover, we can use more related articles for the BERT model, combined with XLNET or Roberta and other popular cutting-edge ideas to improve the BERT model to improve our own training model to obtain a better embedding effect. On the other hand, in the clustering method, we still need to pay attention to improving the clustering model to obtain better clustering results. In turn, it can more accurately and efficiently help classify the literature related to our explosive growth of coronavirus. A better clustering effect means a better focus on researchers' energy. A better clustering effect can also

speed up our server's computing speed and further improve the user experience so that we can also better serve researchers to make our contribution.

Similarly, our website still needs to be improved. The first aspect is the operational aspects. We have not yet passed some security operation and maintenance tests, including stress resistance tests, anti-attack tests, etc. The current website cannot withstand high concurrent visits, and excessive visits may cause server downtime or restricted network traffic. One of the future development directions of the website is to ensure safe operation. Furthermore, due to the current server resource problems, our computing speed is not very fast, so users' time to wait will increase, especially when there are too many concurrent users. How to solve the computing problem concurrently will be faced by the website's back end in the future—a big problem. Therefore, we will use more advanced technology to protect our website so that it can withstand higher concurrent network visits and dynamically manage our server resources so that it can perform operations faster and more stably and better serve the vast number of scientific researchers. At the same time, our website is also considering some new functions, including some advanced query functions, to better target queries or specify keywords for query in different clusters. At the same time, the user feedback policy will also be opened. Users can upload data independently. After manual review, we enriched our database. These functions will be developed after a careful needs assessment. At the same time, we also plan to add some additional features to facilitate users to obtain more accurate results. First of all, we plan to add a form search box. Users can copy the relevant document content and paste it into the corresponding input box (such as title, author, abstract, body, etc.) so as to make more accurate queries more convenient. Simultaneously, we will also communicate with various databases and may directly provide links to related documents for more convenient downloading.

## References

- Devlin, J. , Chang, M. W. , Lee, K. , & Toutanova, K. . (2018). Bert: pre-training of deep bidirectional transformers for language understanding.
- Peters, M. , Neumann, M. , Iyyer, M. , Gardner, M. , & Zettlemoyer, L. . (2018). Deep contextualized word representations.
- Blei, D. M. , Ng, A. Y. , Jordan, M. I. , & Lafferty, J. . (2012). Latent dirichlet allocation. *J. Mach. Learn. Res.* 3, 993-1022.
- Stevens, K., Kegelmeyer, P., Andrzejewski, D., & Buttler, D. (2012, July). Exploring topic coherence over many models and many topics. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning* (pp. 952–961). Jeju Island, Korea: Association for Computational Linguistics. Retrieved from <https://www.aclweb.org/anthology/D12-1087>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Polosukhin, I. (2017). Attention is all you need.
- Wang, L.L., Lo, K., Chandrasekhar, Y., Reas, R., Yang, J., Eide, D., Funk, K., Kinney, R.M., Liu, Z., Merrill, W., Mooney, P., Murdick, D., Rishi, D., Sheehan, J., Shen, Z., Stilson, B., Wade, A., Wang, K., Wilhelm, C., Xie, B., Raymond, D., Weld, D.S., Etzioni, O., & Kohlmeier, S. (2020). CORD-19: The Covid-19 Open Research Dataset. ArXiv.
- Eren, N. N. C. R. E., E. Maksim. Solovyev.(2020, April).Covid-19 literature clustering . . TBA. Retrieved from <https://github.com/MaksimEkin/COVID19-Literature-Clustering>(Malware Research Group
- Dong, Ensheng, Du, Hongru, & Gardner, Lauren. (2020). An interactive web-based dashboard to track COVID-19 in real time. *The Lancet Infectious Diseases*, 20(5), 533-534.
- Beltagy, Iz, Lo, Kyle, & Cohan, Arman. (2019). SciBERT: A Pretrained Language Model for Scientific Text.
- Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao, & Junjie Wu. (2010). Understanding of Internal Clustering Validation Measures. 2010 IEEE International Conference on Data Mining, 911-916.

- Halkidi, M, & Vazirgiannis, M. (2001). Clustering validity assessment: Finding the optimal partitioning of a data set. *Proceedings 2001 IEEE International Conference on Data Mining*, 187-194.
- 唐晓波, & 王洪艳. (2012). 基于潜在语义分析的微博主题挖掘模型研究. *图书情报工作*, 56(24), 114-119.
- 戴天, 吴渝, & 雷大江. (2016). 利用组合模型生成微博热点话题事件摘要. *计算机应用研究*, 33(7), 2026-2029.
- Zengchang Qin, Yonghui Cong, & Tao Wan. (2016). *Topic modeling of Chinese language beyond a bag-of-words*. Academic Press Ltd.
- 王李冬, 张引, & 吕明琪. (2015). 基于词组主题建模的文本语义压缩算法. *西南交通大学学报*, 50(4), 755-763.
- 何伟林, 谢红玲, & 奉国和. (2018). 潜在狄利克雷分布模型研究综述.
- Blei, D. M. , Ng, A. Y. , Jordan, M. I. , & Lafferty, J. . (2012). Latent dirichlet allocation. *J. Mach. Learn. Res*, 3, 993-1022.
- Hofmann, T. . (2017). Probabilistic latent semantic indexing. *Acm Sigir*, 51(2), 211-218.
- Bourlard, H. , & Kamp, Y. . (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4-5), 291-294.
- Hinton, G., E., Salakhutdinov, R., & R. (2006). Reducing the dimensionality of data with neural networks. *Science*.
- Andrew, N. (2011). Sparse autoencoder.
- Liu, M., Shao, Y., Li, R., Wang, Y., Sun, X., Wang, J., & You, Y. (2020). Method for extraction of airborne LiDAR point cloud buildings based on segmentation. *PLoS One*, 15(5), e0232778.