A Dissimilarity-based Concordancer

Abstract

Typical and historical concordancers are similarity-based and return all matching occurrences, or concordances, for a given target word. This allows for a user to observe how a word appears in a corpus, but can be exhaustive, as often all concordances must be manually parsed by the user to be of any use. We investigate and propose an automatic alternative method for returning the most different concordances, based on dissimilarity. We do this by exploring different feature engineering methods for text data, unsupervised learning clustering algorithms, and dimension reduction algorithms. We also propose and demonstrate the use of ROUGE-l as an evaluation metric for dissimilarity, as well as demonstrating optimal methods to improve ROUGE-l scores with the use of a KMeans algorithm that automatically determines the value of 'k' clusters, text sequence representation using Word2Vec for feature engineering, and PCA with a value of 16 for n_components to reduce the features' dimensions effectively.

Introduction

The project goal was to research and implement a new corpus exploration tool. Such a tool can be used to explore the diverse instances where a specific keyword can appear in a given corpus. Current existing technology, such as concordancers and Keyword in Context (KWIC) tools, can provide similar functionalities to our goal but differ in that they return contexts which are similar, whereas our desired functionality is to return contexts that are not similar. Therefore, the primary aim is to build this new functionality which existing concordancers and Keyword In Context (KWIC) tools do not provide.

The key deliverable is a concordancing technique that would employ natural language processing and unsupervised machine learning techniques, which extract a good variety of contexts for a given search word; even in the case of unbalanced usage where most instances of a word appear in very similar sentences. The focus is researching and evaluating the performance of various similarity functions, features, and clustering algorithms. Predefined metrics are used to measure concordance similarity in evaluation.

ABOUT THE CAPSTONE PARTNER

CRIM (Computer Research Institute Montreal) is an applied research center focusing on using advances in information technology to contribute to knowledge transfer from research to industry. The three main verticals at CRIM are: **Analytics and human intelligence, Human related technologies and Software science and technology.**

It helps organizations, primarily small and midsize businesses(SMB's), demystify and gain access to leading-edge technology, such as artificial intelligence, and to efficiently address the technological challenges they face. Its IT researchers and professionals develop a wide array of applications in

diverse areas and work in fields of expertise such as machine learning, computer vision, speech recognition, automatic natural language processing, data science, and operational research. CRIM is a non-profit organization whose neutrality and strong network make it an indispensable resource. Its work is in line with the policies and strategies of its major financial partner, the Ministère de l'Économie et de l'Innovation.

Our mentors Pierre Andre and Lise Rebout are a part of the Speech and Text team and are both experts in natural language processing. They introduced the project to us and together with Prof. Miikka Silfverberg provided constant guidance through weekly meetings.

CONCORDANCER OR Key Word In Context (KWIC)

A concordancer is a tool to show a target word in its context. A concordancer, sometimes called Keyword in Context (KWIC), is a useful tool for linguists, computational linguists and data scientists who need to examine specific occurrences of a target word within a text corpus, referred to as concordances, in order to explore the cues, hints and context that underlie its use. It aligns instances of an expression found in a corpus and shows the surrounding words, enabling the user to assess the usage of the term in the corpus.

The Key Word In Context (KWIC) Indexing technique was based on a previous technique called Keyword in Titles, which was widely used in library science and was a useful method for indexing technical manuals before computerized versions proposed by computer science researchers (H.P Luhn et al, 1960) [1] changed the way concordances were extracted. Concordancers have traditionally been used to provide easy access to key ideas in religious scripts like the Bible, Vedas and Quran to name a few. Essentially, the early concordances would just be an alphabetical list of important words used in a book, listing every occurance of each word with its context. This was done to provide readers with the central themes and ideas of the book or text, with some associated information to help understand the context of the same.

With advances in technology, concordancers are now used in corpus exploration, providing linguists with a tool to study a large corpus by comparing different uses of a term, finding phrases, and examining idioms used in the text. Many large corpora, like the American National Corpus, British National Corpus, or Corpus of Contemporary American English, are released with concordancing features enabled. However, there are many softwares available that can work as stand alone applications that employ concordancing techniques, which are called concordancers.


THE
EXHAUSTIVE CONCORDANCE
OF
THE BIBLE;

Figure 1. A traditional concordancer

APPLICATIONS OF CONCORDANCERS

Concordancers are vital tools in library science for indexing texts, popular in religious domains. They aim to count and define the usage of a word by its occurrence, to provide an insight of word structure in authentic texts. Additionally, concordancers also have applications in English Language Teaching (ELT) , where they are used to acquaint learners with collocations and word usage. Using concordances, students can learn the frequency of use and the context in which words should be used (Fatih 2014) [3]. Furthermore, within the field of computational linguistics, concordancers can be used to explore corpuses and enable researchers to quickly analyse a large corpus for frequency of n-grams and corpus diversity.

CURRENT CONCORDANCING TOOLS

The Natural Language Toolkit (NLTK) is a widely used suite of python libraries and other resources that is a popular platform used for natural language and text applications. The NLTK suite also has a concordancing functionality for the in-built corpora distributed by the framework. It provides the concordance function to show a snippet of text, outlining every occurrence of a given word, along with some context around it. It does the same by extracting the occurrences using a concordance index of the pre defined corpus and returning all the instances as seen in the corpus.

```
>>> text1.concordance("monstrous")
Displaying 11 of 11 matches:
ong the former , one was of a most monstrous size . ... This came towards us ,
ON OF THE PSALMS . " Touching that monstrous bulk of the whale or ork we have r
ll over with a heathenish array of monstrous clubs and spears . Some were thick
d as you gazed , and wondered what monstrous cannibal and savage could ever hav
that has survived the flood ; most monstrous and most mountainous ! That Himmal
they might scout at Moby Dick as a monstrous fable , or still worse and more de
th of Radney .'" CHAPTER 55 Of the monstrous Pictures of Whales . I shall ere l
ing Scenes . In connexion with the monstrous pictures of whales , I am strongly
ere to enter upon those still more monstrous stories of them which are to be fo
ght have been rummaged out of this monstrous cabinet there is no telling . But
of Whale - Bones ; for Whales of a monstrous size are oftentimes cast up dead u
```

Figure 2. Output of NLTK Concordance function

Apart from corpora with concordancing features, there are many open source concordancing softwares available that can display concordances for any corpus being processed through it; provided it has the required format.

Some of the more prominent open source concordancers are AntConc, WordStatix, and Simple Concordance Program, for general text concordancing. There are also some concordancers like AV Bible and Interlinear Scripture Analyzer, which are tailored for use in the specific domain of religious texts.

Figure 3. A snapshot of AntConc

One of the features that these open source concordancers have is that the concordances returned are in order of occurrence of the instance. Therefore, it is possible that all or a vast majority of the concordances returned showcase the query word in the same ‘sense’ or context.

A more ideal concordance output would highlight all the captured senses or contexts that a word has been used in and return concordances in a way that captures and highlights this diversity.

There have been efforts made in the direction of including sense or contextual information to concordance outputs in the past, but they relied heavily on lexical analysis performed by linguists while annotating data. One such approach was the semantic concordancer, which annotated the concordances of a query word with the corresponding WordNet synset that captured the sense in which the word had been used. (Miller et al, 1993) [3]

Data

THE ASSNAT AND COVID

Since the goal of this project was building a bilingual concordancer, the CRIM team provided two text corpora, the Assnat and COVID-19 datasets. The Assnat dataset is a French language dataset which is collected from Travaux de l’assemblée nationale. Travaux de l’assemblée nationale means “Work of the National Assembly”, in English. This official website offers searching and downloading services that allow users to retrieve documents of the parliamentary work. Parliamentary work basically refers to the work, discussion and debate of the Assembly and of parliamentary committees. Specifically, this dataset is a collection of literal recordings or transcripts of parliamentary work in Quebec from 1908 to 2019. The contexts of these raw texts are various, such as culture and education, agriculture, fisheries, energy and natural resources, economics and work, and citizen relationships. Another dataset is the COVID-19 Open Research Dataset which is an English language corpus. The

COVID-19 dataset is built and released by the Semantic Scholar team at the Allen Institute for AI. It is a free and open resource which is built upon more than 59,000 scholarly articles related to the novel coronavirus. Obviously, this text corpus is related to biology, pathology and virology. More specifically, the contexts are revolving around the coronavirus in 2019. Primarily, the two datasets contrast not only in language, but also in domain and periodicity.

THE STRUCTURE OF DATASETS

Both of these corpora are organized under the same structure. Below is a dummy example.

```
- corpus.json
- CorpusStructure.json
- documents
        - 0a01ca28-7ebc-11ea-b2fa-02420a0000bf.json
        - 0a1e631c-7ebc-11ea-9100-02420a0000bf.json
        - ...
 - groups
        - 757fc491-0f25-4ad7-914c-cd906afa7f13
            - 0a01ca28-7ebc-11ea-b2fa-02420a0000bf.json
            - 0a1e631c-7ebc-11ea-9100-02420a0000bf.json
            - ...
        - 2b830e5f-f607-4ff8-b38e-3a4d028cced8
            - 0a01ca28-7ebc-11ea-b2fa-02420a0000bf.json
            - 0a1e631c-7ebc-11ea-9100-02420a0000bf.json
            - ...
```

Figure 4. Structure of the corpus

For each corpus, there are two description json files, corpus.json and CorpusStructure.json, as well as two folders, documents and groups.

corpus.json


```
{
    "id": "26188747-d7d5-401d-8414-f48cee2da491",
    "title": "AssNatBase",
    "languages": [
        "fr-FR"
    ],
    "projects": [],
    "creationDate": "2020-04-07T19:56:02.000Z",
    "lastModifiedDate": "2020-04-09T20:42:58.000Z",
    "documentCount": 5897
}
```


Figure 5. corpus.json

corpus.json is mainly describing the information of this corpus which contains a unique corpus id, corpus title, language, creation date, last modified data and the count of documents in this corpus.

CorpusStructure.json


```
{
    "buckets": [
        {
            "id": "757fc491-0f25-4ad7-914c-cd906afa7f13",
            "name": "Transcode task bucket",
            "schemas": [
                {
                    "schemaType": "DOCUMENT_META",
                    "jsonSchema": null
                }
            ]
        },
        {
            "id": "2b830e5f-f607-4ff8-b38e-3a4d028cced8",
            "name": "Annotations",
            "schemas": [
                {
                    "schemaType": "SENTENCE",
                    "jsonSchema": null
                },
                {
                    "schemaType": "TOKEN",
                    "jsonSchema": null
                },
            ]
        }
    ]
}
```


Figure 6. CorpusStructure.json

CorpusStructure.json contains detailed information of raw text documents along with their annotations and explicates the structure of ‘groups’ folder. As shown in the example above, there are two buckets here, which means there will be two subdirectories under ‘groups’ folder.

groups

Via matching the bucket ids, we can observe the corresponding subdirectories under the ‘groups’ folder. The first bucket is the ‘Transcode’ task bucket, which stores the metadata of each document.

```
{
    "26188747-d7d5-401d-8414-f48cee2da491":
    {
        "757fc491-0f25-4ad7-914c-cd906afa7f13":
        {
            "DOCUMENT_META":
            [
                {
                    "schemaType": "DOCUMENT_META",
                    "_documentID": "0a2b03de-7a9b-11ea-8913-02420a0000bf",
                    "_corpusID": "26188747-d7d5-401d-8414-f48cee2da491",
                    "file_name": "39-1_20100216.txt",
                    "file_path": "/brut/39-1/",
                    "file_type": "text/plain; charset=UTF-8",
                    "file_encoding": "UTF-8",
                    "source": "brut.zip",
                    "indexedLanguage": "fr-FR",
                    "detectedLanguage": "fr-FR",
                    "detectedLanguageProb": 99.99974086418732,
                    "file_creation_date": "2020-04-07T15:43:43Z",
                    "file_edit_date": "2020-04-02T18:06:48Z",
                    "document_size": 269574,
                    "file_size": 269573,
                    "file_extension": ".txt",
                    "annotationId": "0a4ff60a-7a9b-11ea-a2f1-02420a0000bf"
                }
            ]
        }
    }
}
```


Figure 6. ‘Transcode’ task bucket

The metadata includes schema type, document id, corpus id, file name, file path, file type, file encoding, source, indexed language, detected language, file size, file extension, annotation id and so on.

Another bucket is the ‘Annotations’ that contains information of all possible schemas.

```
{
    "26188747-d7d5-401d-8414-f48cee2da491":
    {
        "2b830e5f-f607-4ff8-b38e-3a4d028cced8":
        {
            "SENTENCE":
            [
                {
                    "schemaType": "SENTENCE",
                    "_documentID": "0a2b03de-7a9b-11ea-8913-02420a0000bf",
             "offsets": [{'begin': 0, 'end': 107}],
                    "string": "Treize heures quarante-six minutes) Le Vice-Président
(M. Chagnon): Bon début de semaine, bon mardi matin.",
                    "_corpusID": 26188747-d7d5-401d-8414-f48cee2da491,
                    "length": 107,
                    "annotationId": ba98b970-8542-11ea-8423-02420a00008d
                },
                ...
            ],
            "TOKEN":
            [
                {
                    "_documentID": "0a2b03de-7a9b-11ea-8913-02420a0000bf",
                    "category": "PUN",
                    "string": "(",
                    "offsets": [{'begin': 0, 'end': 1}],
                    "length": 1,
                    "schemaType": "TOKEN",
                    "stem": "(",
                    "lemma": "(",
                    "_corpusID": "26188747-d7d5-401d-8414-f48cee2da491",
                    "annotationId": "baa5a03a-8542-11ea-8ccc-02420a00008d"
                },
                ...
            ]
        }
    }
}
```


Figure 7. ‘Annotation’ task bucket

For both the French and English language corpora, the annotation files mainly have two schema types, ‘SENTENCE’ and ‘TOKEN’. These two schemas represent two types of annotations, split sentences and tokenized words. The sentence field consists of a list of dictionaries where it saves information like schema type, document id, offsets, string, corpus id, sentence length and a unique annotation id. The token field is a list of dictionaries where it has document type, category which means the part-of-speech of this token, string, offsets, length of this token, schema type, stemmed token, lemmatized token, corpus id and annotation id.

documents

Finally, raw text documents are in the ‘documents’ directory where each json file maintains the whole information of a single document.

```
{
    "id": "0a2b03de-7a9b-11ea-8913-02420a0000bf",
    "source": "brut.zip",
    "title": "39-1_20100216.txt",
    "text": "(Treize heures quarante-six minutes)
Le Vice-Président (M. Chagnon): Bon début de semaine, bon mardi matin. Veuillez vous
asseoir, s'il vous plaît.
Affaires courantes Déclarations de députés
Nous allons passer immédiatement à la rubrique Déclarations des députés, et je vais
inviter M. le député de Charlesbourg à prendre la parole. ...",
    "language": "fr-FR"
}
```


Figure 8. one raw document file

Basically one json file includes an id, source, title of the document, full text and the language.

Table 1. Statistics information about corpora

datasets	name	language	document count	token count	Annotated or not
Travaux de l'assemblée nationale	AssnatBase	french	5,484	231M	yes
COVID-19 Open Research Dataset	Covid19Txt	english	57,368	295M	yes

As Table 1. shows, there are over 5 thousands documents and over 231 million tokens in the Assnat corpus. And there are 57 thousands documents in the Covid corpus. It should be noticed that both corpora are quite big and take around 80G space after decompression. In this project, the input data are query words and their contexts. Considering huge sizes of these corpora, searching all occurrences of a target word will be time-consuming if we use a simple linear searching function.

Methods

The whole project can be divided into 4 stages as illustrated below:

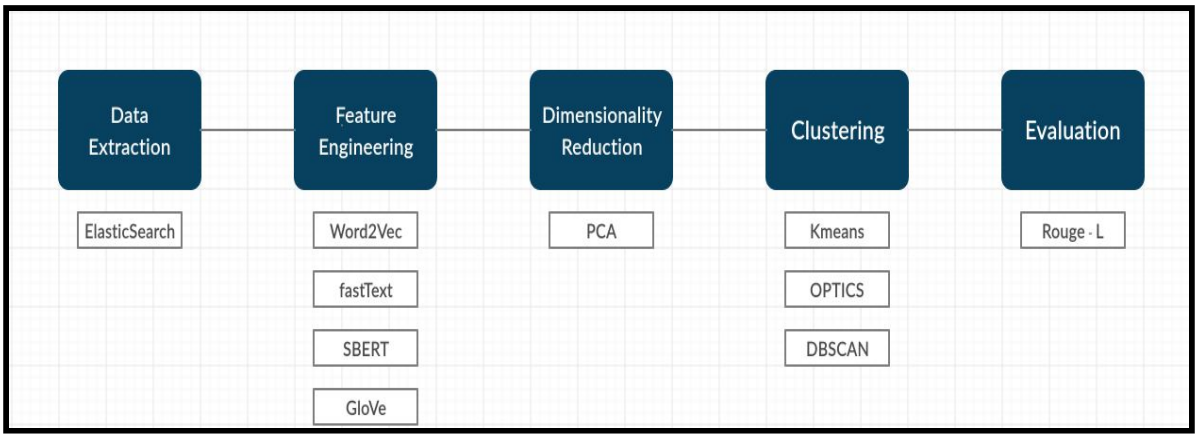


Figure 9. Pipeline of this project

Phase 1. Data Extraction

Since the data for the project comprises two large corpuses, COVID-19 (English language) and Assnat (French language), the first step of data retrieval was a challenge. This was because the simple python implementation of a search function to retrieve contexts, containing the query word, would take an incredibly long time to execute which makes the approach inappropriate for use. Thus, to reduce the time required to retrieve contexts, we implemented the elasticsearch framework. Elasticsearch is an open source distributed, RESTful search and analytics engine capable of efficiently solving search and querying for large datasets.[4]

The inverted index feature of elasticsearch framework is the reason why querying operations are so speedy and efficient. In layman terms, when a document is fed to the elastic search cluster, it is lemmatized, tokenized and all the stop words are removed. Once this is done, each unique token is stored as a key which points to a collection of documents in which the term has occurred. The inverted index created is stored in buffers and eventually is compressed to be held in data structures called segments. Each segment consists of inverted indices and a collection of the same make up another structure known as a shard.

Once the entire data is indexed and stored as immutable segments in shards, the shards are then used for querying. The real strength of elasticsearch comes from it's indexing features, which are used to create a json store instead of a traditional database. This collection of json objects is stored on an Apache Lucene search engine which ultimately decides the indexing rules which would increase efficiency and decrease latency.

The code implementation of the same can be found in the **corpus_reader.py** and **initialize_index.py** files under the source folder of our directory.

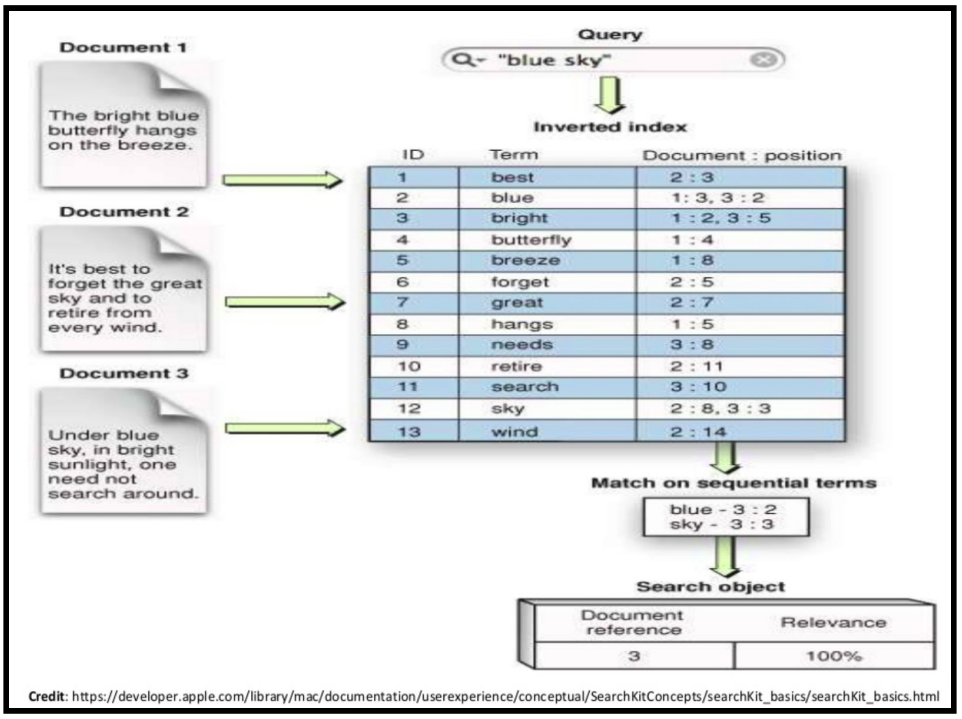


Figure 10. Procedure of Elasticsearch indexing

For our project, we have created two elastic search indices, one for each corpus. These indices are then integrated into our search functionality where we query the nodes in order to retrieve the contexts which contain the query word. More specifically, the search function to takes an argument: the query word , the window size (i.e the number of tokens before and after the query word, which is the context), and the part of speech tag of the query word to ensure that we capture the right sense of the word.

The search function returns a list of triples of the following structure:
[([LEFT CONTEXT TOKENS], target word, [RIGHT CONTEXT TOKENS]), ...]

This list is then used for further processing in our project pipeline.

Phase 2. Feature Engineering

This phase of the pipeline consists of deriving meaningful features from the contexts / instances retrieved from the elasticsearch querying. The goal is to vectorize this collection of instances so that each instance can be converted into a vector representation. Word vectorization is a process of converting text data into numbers or vectors. Typically this process utilizes a deep learning algorithm and gradient descent, in order to optimize a given loss function; however, the specifics are related to which feature engineering method is used. Often, embeddings are created by a combination of skip gram and CBOW approaches to learn appropriate vectors for words based on their frequency, position, and the words surrounding a given word to be vectorized. The context of a given query word encapsulates information about the meaning and usage of the word. Since this process can be computationally expensive and time consuming, models trained on outside data sets can be imported and utilized through transfer learning.

The vectors can then be used to perform syntactic and semantic comparison using metrics like cosine similarity and euclidean distance between a pair of vectors.

In our implementation we consider each context to be an aggregated vector which is formed from the vectors of the words found in the context. To explore the efficiency of word embeddings, we chose four techniques to extract vectors. The code implementation can be found in the **feature_extractor.py** file in the source folder of our directory.

1. Word2Vec

For our implementation we have made use of the gensim library to extract word embeddings. First we use the list of tokens from a corpus and create one English language and one French language word2vec model using these tokens, which was derived from 10,000 documents in COVID and 1000 documents from Assnat corpus.

Once the model has been created, we then extract vectors for each word in each instance. This is done by performing a lookup in the word2vec vocabulary. All out-of-vocabulary instances are handled by assigning an aggregate vector to such keywords. The aggregate vector as the name suggests is simply a vector mean of all the vectors contained in the word2vec model.

The Word2Vec technique is used in the **word2vec function** of our code. The function's input is a list of instances and the word2vec model and it returns two arrays of vectors: left_context and right_context. Each context representation is a single vector, which is the average of all word vectors within that context. The two context vectors returned are later concatenated together.

The query word itself is not considered when creating a vector representation since it will be common amongst all the instances and can lead to increased similarity between instances if included in the vectorization scheme.

2. GloVe

GloVe is similar to Word2Vec in that it is token based, however while Word2Vec uses local statistics to minimize a loss function on predicting the next token, GloVe uses global statistics. Our implementation of GloVe is very similar to Word2Vec, except that we do not further train GloVe on our own corpora and instead use the pretrained 50 dimensional word embeddings provided by the GloVe developers. [1]

Just like our **word2vec function**, we have a **glove function** which is nearly identical in architecture except that the Word2Vec model has a unique class object, whereas GloVe embeddings are stored in a txt file, which is later stored as a variable as a python dictionary.

3. fastText

fastText is a word embeddings library developed by Facebook. [6] The word vectors have been trained on a corpus of text extracted from wikipedia. fastText provides multilingual word embeddings which is a feature we used to extract vectors for both our English and French language corpuses.

[1] GloVe embeddings can be found here

The advantage of using fastText is that not only does it provide better representations due to transfer learning but it creates word vectors based on character level information. Character level implementation means that a given word is first broken down into n-grams and the word vectors are then created using the individual representations of these character level n-grams.

This character level embedding functionality is very beneficial since it eliminates the out of vocabulary problem that other techniques like Word2Vec suffer from.

We make use of pre-trained english and french word embeddings as provided by fastText The fastText implementation can be found in the **fasttext function** of our code. We make use of the SISTER[2] package which helps reduce the time spent in downloading and reading from the fastText vector files. We are not retraining the fastText model on our corpus data for convenience, instead we are simply using a lookup to extract the vectors from pre-trained embeddings shared by fastText.

4. SBERT (Sentence BERT)

The BERT architecture is the current de facto standard in creating semantically meaningful word embeddings. The SBERT method uses siamese or twin BERT models to classify whether two sentences are similar or not.[5] The word embeddings themselves have been created by training the SBERT model on some of the industry standard datasets used for semantic evaluation, such as the Stanford Natural Language Inference dataset and MultiGenre - NLI dataset. The SBERT model has been proven to give excellent results on semantic tasks such as Semantic Textual Similarity and SemEval tasks.

Our implementation of SBERT can be seen in the **sbert function** of our code, which receives an input of an array of instances and returns the vectorized left and right contexts.While the SBERT method generates superior word embeddings, a key shortcoming is that SBERT is trained on complete sentences but the data at hand is are strings capturing the left and right context of a given query word.Therefore the SBERT generated embeddings are heavily dependent on the window size used to capture left and right context. A smaller window size may lead to inaccurate or inefficient word embeddings.

Phase 3. Dimensionality Reduction

Many clustering algorithms suffer from the curse of dimensionality and word embeddings can typically have 256, 512 or 1024 dimensions, therefore we introduced dimensionality reduction before clustering.

The goal of dimensionality reduction is to reduce the complexity of a machine learning model and to avoid overfitting. In our case we want to reduce the dimensionality of the word embeddings produced by the vectorization techniques.

---

[2] SISTER package is an open source PyPI package that helps streamline the download and read process for fastText .vec files https://pypi.org/project/sister/

The code implementation of dimensionality reduction can be seen in the **extract_features** function of the feature_extractor.py file in the source folder of the repository.

- ***PCA (Principal Component Analysis)***

As the name suggests we use this technique to compress the multidimensional vectors generated in the previous step into a lower-dimensional feature subspace , with the goal of maintaining the most relevant features/dimensions.[8]. The goal is to produce a set of selected dimensions or features, which can explain a large percentage of the variance as observed in the original higher dimension representation.

PCA uses correlations to find patterns in the data. It then finds the directions of maximum variance in the higher dimension and projects this variance into a lower dimension , thus preserving all the explainable variance while reducing the number of dimensions of the word embeddings.

- ***t-SNE***

t-SNE or t - distributed stochastic neighbor embedding is a popular visualization technique used for word embeddings. t-SNE is a non-linear dimensionality reduction technique , since it is non-linear it can adapt to underlying data and use different transformations for different regions of the data.[9] t-SNE is most commonly used for visualizing word embeddings in 1D or 2D.

While PCA is a parameter free approach, t-SNE has several hyperparameters (initial_dims, perplexity and max_iter) which can help extract better dimensions from the input data. For our purposes we have used the scikit learn default hyperparameter.

Phase 4. Clustering

Once we have converted all the instances or sentences in the corpus that contain the query word into vectors, we can then use unsupervised clustering to group instances with similar meanings together.

The intuition behind clustering being that instances that use the query word in the same sense or context , would generate vectors which are closer together and vice versa. For our clustering implementation we used three different clustering algorithms : KMeans , OPTICS and DBSCAN.

- ***KMeans***

KMeans clustering is an unsupervised approach where the vectorized data is split into ‘k’ clusters according to a distance or similarity metric. The ideal ‘k’ number of clusters would then ensure that all instances inside a cluster are similar and that each cluster is at maximum distance from the other clusters i.e they are far apart or dissimilar.[7]

The most challenging hyperparameter for KMeans clustering to give good results is the number of clusters or ‘k’. Since in the present case there is no predefined number of clusters for any query word, we use the kmeans inertia plot to find the optimal value of ‘k’.

The inertia attribute of the KMeans algorithm measures the sum of squared distances that each observation has between itself and the nearest cluster. This inertia metric can then be used to guide on what the optimal value for ‘k’ would be. If one were to consider a situation where ‘k’ is equal to the number of observations then the inertia would be zero and in the opposite case where ‘k’ has a small value like one or two inertia would be very high. Therefore to find the optimal value of ‘k’ , one has to find the elbow of the inertia plot, which can be considered as the point in which the trend of the distribution begins to approach linearity and flatten out. Following this logic, in Fig 11 below one may consider the elbow to be at the k = 3.

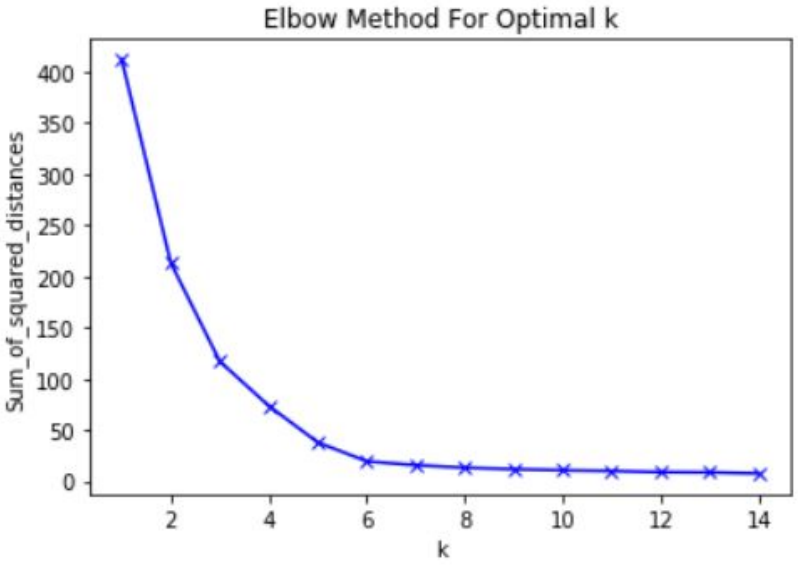


Figure 11. Elbow method for picking an optimal K

The code implementation of the same can be seen in the **kmeans** and **avg_inertia_elbow** functions in the cluster.py file in the source directory. Where we developed an automatic method for determining the elbow or the optimal ‘K’ value by using a distribution of second order derivatives, and choosing the point whose second order derivative is closest to the mean second derivative. Since there can be random variation from the KMeans algorithm, the distribution of inertia can also vary, which is why the **avg_intertia_elbow** function allows for a user to run several instances of KMeans and average the results to apply a smoothing effect to the inertia distribution.

- ***DBSCAN***

DBSCAN or Density Based Spatial Clustering of Applications with Noise is a distance measure based clustering technique [11] . DBSCAN is a good choice in situations where it is difficult to estimate the number of clusters the data can have. It works to cluster the data based on two hyperparameters:

1. Epsilon
   This hyperparameter determines how close two points should be to each other in order to be considered as a part of the same cluster, measured in Euclidean distance. By default we set the epsilon value to be 30 in our approach. To tune epsilon, one can consider visualizing an ordered plot of each data points’ distance to their nearest neighbors, and choose the distance at which there is maximum curvature after the graph’s most horizontal trend, such as seen in Fig 22.

2. MinPoints
   This hyperparameter sets the minimum number of samples each collection should have in order to be considered as a cluster. The default value of minPoints in our implementation is 2, which was more or less determined through trial and error.

   Even though DBSCAN helps us avoid the problem of having to determine the optimal number of clusters 'k', the values of the two hyperparameters still pose a challenge since they can affect the performance of our clustering operation.

The code implementation for DBSCAN can be found in the **dbscan** function of clustering.py.

- ***OPTICS***

OPTICS also called Ordering Points to Identify Cluster Structure, this technique is derived from the DBSCAN clustering fundamentals.Unlike KMeans and DBSCAN, OPTICS clustering doesn't heavily rely on setting hyperparameters and it has been considered to be a parameter less clustering technique in some situations [10]. Since OPTICS is still a derivative of DBSCAN the hyperparameter min samples remains. But the way OPTICS works is that using the value set for min samples, it calculates a core distance and reachability distance for each sample in the dataset and updates the cluster centers according to this.

Due to this new framework of analysing each sample with respect to its distance from the rest of the cluster structure, OPTICS can do a very good job at identifying and isolating outliers which could otherwise affect the clustering scheme.

In our implementation of OPTICS, we set min samples to 2, using the same method as we did with DBSCAN, and we have also used cosine similarity to be the guiding metric for clustering. The code for the same can be found in the **optics** function of **cluster.py**, in the source folder.

Evaluation

After clustering the word embeddings using any of the above mentioned techniques, we extract the cluster centers or take random samples from each cluster as per clustering technique. This new set of samples is then treated as our final context diverse concordance result for a given query word.

- ***ROUGE SCORE***

To evaluate the context diversity of the concordances returned after clustering we use a metric called ROUGE (Recall Oriented Understudy for Gisting Evaluation) [12]. A meric commonly used for the evaluation of automatic summarization tasks to compute the similarity between the original text and the computer generated summary of the same.

$$ROUGE - N = \frac{\sum_{S \in S_H} \sum_{g_n \in S} C_m(g_n)}{\sum_{S \in S_H} \sum_{g_n \in S} C(g_n)}$$

where

S_H is the set of manual summaries

S is an individual manual summary

g_n is an N-gram

$C(g_n)$ is the number of co-occurrences of g_n in the manual summary and automatic summary

Figure 12. Rouge Score

The Rouge metric is calculated by counting the ngram overlaps occurred between two text samples as shown in Fig 12. Therefore if we were to calculate a ROUGE score of two identical text samples it would be one and the more dissimilar the text samples, the lower the ROUGE score will be.

We use this intuition to calculate the ROUGE score of the concordances produced after clustering. We would expect the ROUGE score before clustering to be higher than the ROUGE score produced after clustering and in general for any given concordance output we would like to observe lower ROUGE scores to indicate that the samples are context diverse. In other terms we expect the ROUGE score for cluster centroids to be lower than the same for randomly picked samples.

In our implementation of the ROUGE score evaluation we use the ROUGE - LCS variant, LCS standing for lowest common substring. The reasoning behind it being that if two text samples were to have the same phrase or LCS then it's most likely that they have used the query word in the same context.

Since our concordance results consist of multiple samples, we compute a pairwise ROUGE score for each concordance output. Since the resulting metric represents a distribution we have included the min, max and standard deviation observed for each concordance result in order to give a better insight to the similarity observed between the samples of a given concordance output.

- ***COSINE SCORE***

Cosine distance is a classic metric used for evaluating the similarity. For cosine distance, greater the distance is, less similar between two vectors. In our experiments, the KMeans algorithm uses cosine metric during each update step. So for the sanity test, we included cosine score as an alternative metric.

Same as calculating ROUGE score, the first step was pairing the concordance results. And then we averaged the cosine distances of each pair as our cosine score. There is a difference from ROUGE score that the bigger the cosine score is, the better the concordancer is. During the feature extraction phase, we could get the vectorized results which we called the full version. Following this phase was PCA where the dimension of vectors would be reduced so that we called the reduced version. According to different versions, we would have two types of cosine scores.

In terms of reduced version cosine score, clustering results are supposed to achieve a better score than random results. Meanwhile we would calculate cosine score in full version as a contrast.

The ROUGE and COSINE implementations have been included in the **evaluator.py** file in the source folder.

Analysis

Comparison between DBSCAN, OPTICS and KMeans

Different clustering algorithms have their own properties. For instance, Kmeans requires a specific number ‘k’ clusters and is very sensitive to noise, while DBSCAN/OPTICS could automatically decide the number of clusters and discard data points which could not be classified to any class. Therefore, it is worth comparing these three algorithms in terms of their performance on both English and French corpora and exploring how other parameters would affect them.

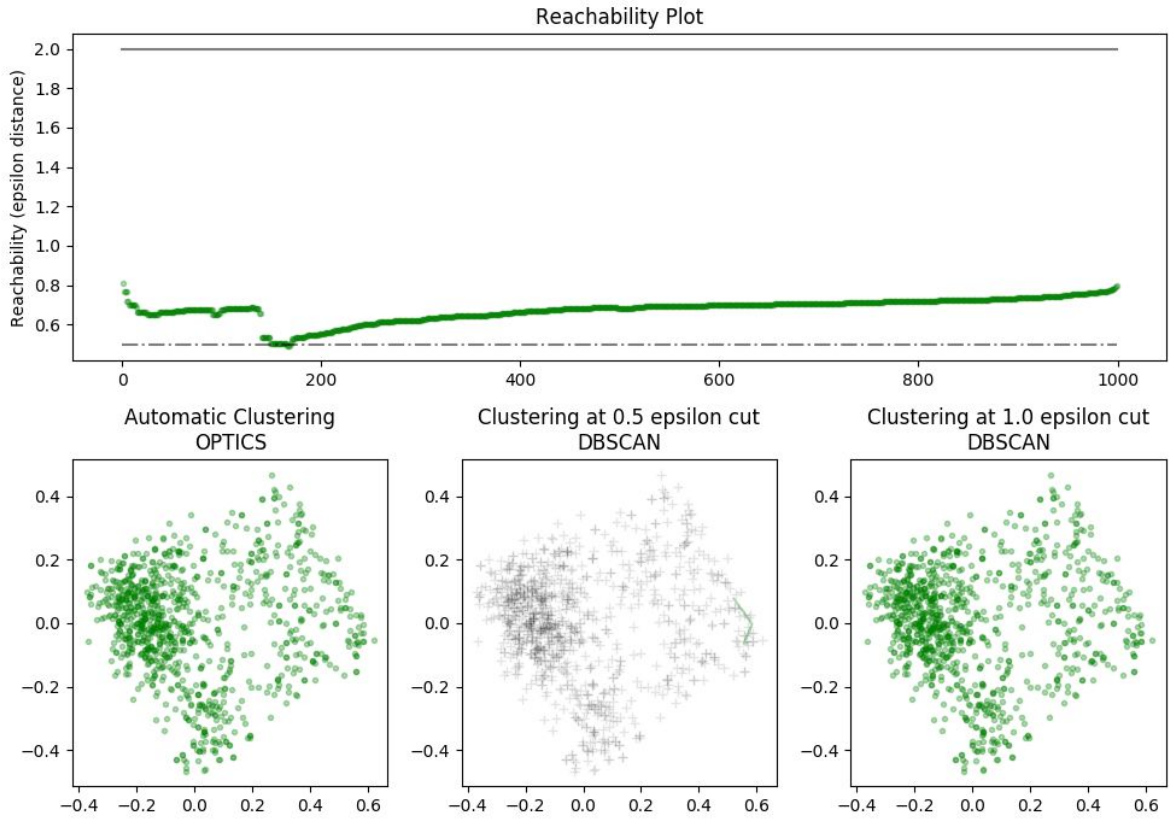


Figure 13. OPTICS and DBSCAN clusters for 'PAUCITY'

Figure 13 is analyzing OPTICS and DBSCAN when testing the word, ‘PAUCITY’. The top plot is reachability plot where reachability means the distance from a data point to the core point in this cluster. In this figure, green dots were reachable data points and were classified into one cluster. Roughly, the reachability value of each point was below 1.0 epsilon distance. In other words, these

data points could be connected to a core point with 1.0 distance. By looking at the first sub-plot on the bottom, OPTICS classified all data into one cluster which was corresponding to the reachability plot. Switching to DBSCAN at 0.5 epsilon, all points were colored gray meaning that they were noise. So there was no cluster given 0.5 epsilon. However, when epsilon was 1.0, all the data points were in one cluster which was exactly the same as OPTICS. In fact, OPTICS's Xi method can account for different choices of eps in DBSCAN. OPTICS/DBSCAN are not sensitive to noise. But in this case, they tend to ignore a great number of points and are sensitive to epsilon.

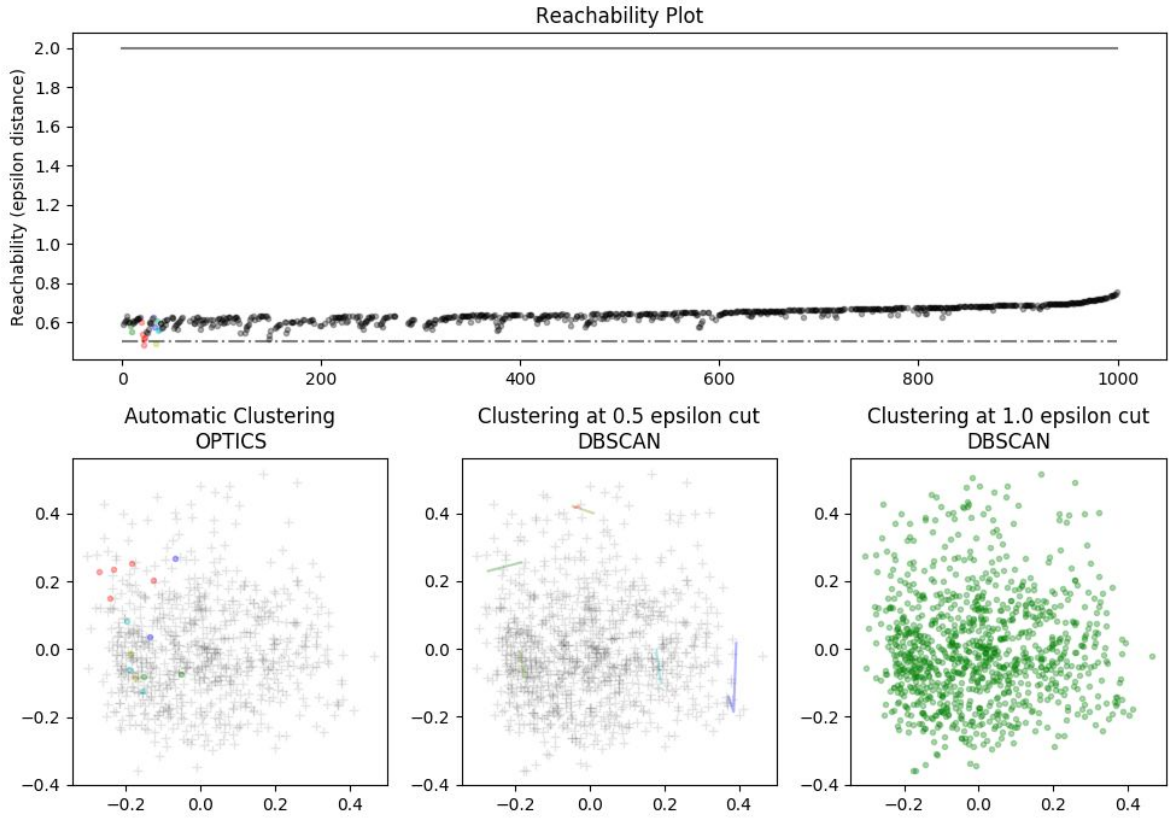


Figure 14. OPTICS and DBSCAN clusters for 'INFECTION'

Figure 14 is another example for comparing OPTICS and DBSCAN. It is a demonstration of clustering results for the query word, 'INFECTION'. Looking at the first sub-plot on the bottom, this time OPTICS threw away tons of points (the gray points) and only few of them(colored points) are clustered. Since only dozens of points were labelled, each cluster might not contain sufficient samples, like 5 data points in the pink cluster. For the DBSCAN method, when epsilon was 0.5, the majority of points were regarded as outliers and would be discarded. In contrast, when epsilon was 1.0, all points were in one cluster. Figure 13 and 14 proves that OPTICS and DBSCAN are not adaptive in this application because they badly depend on the parameters.

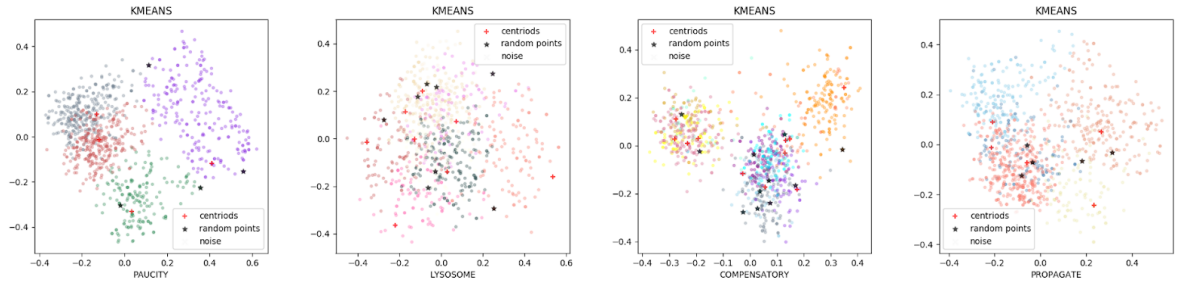


Figure 15. KMeans clusters on low frequency words

Now we are analyzing the KMeans method where we applied the PCA dimension of 2. As we can see from Figure 15, Kmeans would not ignore any point no matter noise or not and tries to classify all of the data points. For the word 'PAUCITY', there are several distinctive clusters segmented by clear boundaries. However for the word 'COMPENSATORY', three big clusters are divided into many messy sub-clusters. After clustering, we would pick centroids of each cluster as outputs. The centroids were marked as red '+'. Meanwhile we marked randomly picked results as black '*'.

Table 3. KMeans results on low frequency words

Query Word	Results	Rouge Score
PAUCITY	basis , there is a PAUCITY of data on the long-term visits Initially the by , PAUCITY of data to level values a s , paralleling the PAUCITY of melanomas reported in other report in 2 , the PAUCITY has of to experimental all	0.0185
LYSOSOME	blood vessel walls which induce LYSOSOME enzyme liberation from neutrophils ■ hydrolyzed by proteases in the LYSOSOME , antigen fragments generated by autophagosome subsequently fuses with a LYSOSOME , enabling the intra-autophagosomal components directly taken up by the LYSOSOME , either through invagination of . to Biol the 17(4) LYSOSOME e2007044 . begins with packaging of the substrates into the LYSOSOME [107] . The it phagosome directly capture data LYSOSOME , a in that instruments the mode of administration of LYSOSOME targeted medica tions in order	0.0039
COMPENSATORY	a p-neighbor differs by a COMPENSATORY base pair-mutation , see Figure is likely due to the COMPENSATORY nature of the hydrophobic interface bZIP60 may serve as a COMPENSATORY mechanism required for the host develop to , the if COMPENSATORY a develop microenvironment cancer increased of β-microglobulins could be a COMPENSATORY reaction to toxic damage , whereas in T3 , the COMPENSATORY changes were both in DNA from the population or require COMPENSATORY mutations . c | A the company would make a COMPENSATORY payment from a " health e.g. , the upregulation of COMPENSATORY pathways when clathrin-mediated endocytosis is seizure free . He utilized COMPENSATORY coping strategies at work due genetic verification , particularly through COMPENSATORY mutagenesis , will be needed	0.0058
PROPAGATE	E6 plate were used at PROPAGATE 37°C MERS-CoV h were followed March 2003 are difficult to PROPAGATE in cell cultures . Their example , of viruses that PROPAGATE by modifying behaviors , although trans . serves NS3 and PROPAGATE were the required viral cis treated 19 0.5 pretreatment to PROPAGATE could form in massive syncytia	0.0

As we can see the results of 'PAUCITY' in Table 3., `there is a PAUCITY of data on the long-term` and `visits Initially the by , PAUCITY of data to level values` should be categorized to the same cluster because both are talking about the lack of data. However, the distant tokens in left and right contexts seem to confuse KMeans. It is likely the SBERT features giving more attention to tokens like 'visits', 'long-term' or 'values' which leads to adverse directions of clustering.

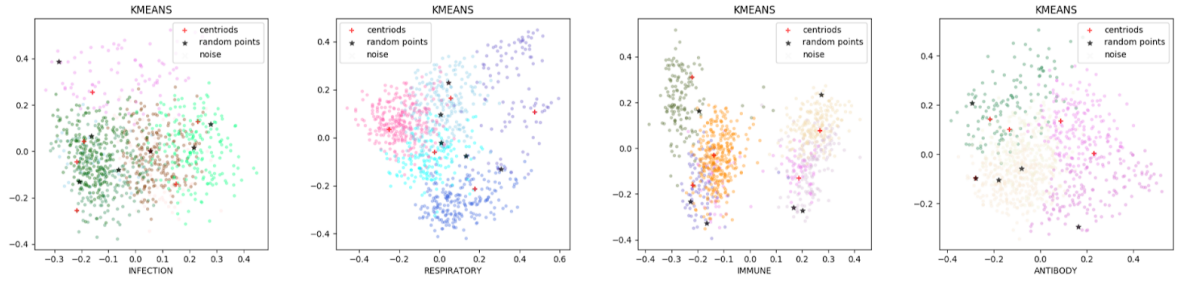


Figure 16. KMeans clusters on high frequency words

Given figure 16, there is no big difference between clusters of high frequency words and low frequency words.

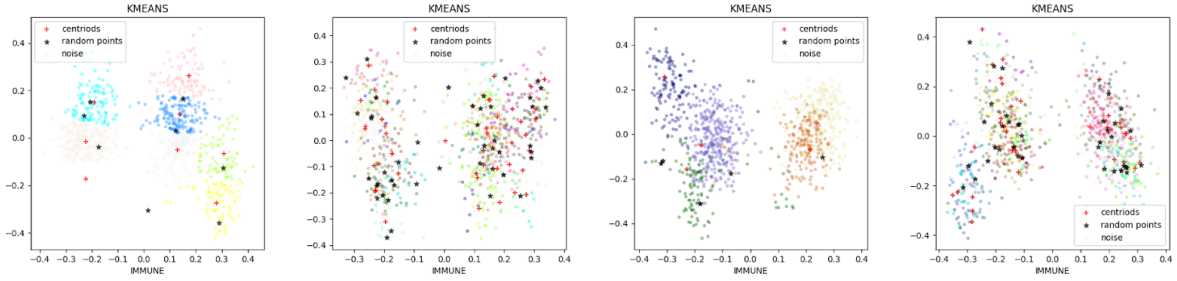
Table 4. KMeans results on high frequency words

Query Word	Results	Rouge Score
INFECTION	antibiotic prescribing . • Wound INFECTION of incised skin or soft such as measures to improve INFECTION control , development of targeted over the last decade , INFECTION with STDs has risen due ZIKV with in NS5 America INFECTION its validated to number put 2-week intervals has also eliminated INFECTION (Baker , 2007 to update the source of INFECTION for case i. The source the context of invasive adenovirus INFECTION . While data on optimal	0.0
RESPIRATORY	parainfluenza-3 they , be problem RESPIRATORY dairymen viruses some BVD-MD , in adults is an acute RESPIRATORY disease that requires invasive mechanical periods . We used all-cause RESPIRATORY infection diagnoses as a proxy in Singapore of severe acute RESPIRATORY syndrome-associated coronavirus (SARS-CoV) during worlds winter however Lina RESPIRATORY al were in detected France	0.0
IMMUNE	oral administration only elicited an IMMUNE response in one of six signaling crosstalk that programs the IMMUNE response to virus infection . of autoantigenic material to the IMMUNE system but , again , For example , when endogenous IMMUNE cells lose their ability to of also IBV induce sequences IMMUNE all Asian countries excluding China	0.0667
ANTIBODY	of cattle induce a strong ANTIBODY response , the time delay in our view . Once ANTIBODY coating proceeds beyond a critical risk passive of transfer = ANTIBODY vaccinated a at protective vaccinee unvaccinated with 1:3000 dilution of anti-His-tag ANTIBODY (Invitrogen) in PBS-T antibody cDNA . The obtained ANTIBODY cDNA is inserted into expression	0.0

Dimensionality Reduction

The initial dimension of Sbert is 1024 which will cause costly calculation for further processing. Another reason for applying dimensionality reduction is that KMeans optimizes variance while in

high dimension, so the impact of neighbours' distance will be weakened. Therefore, dimensionality reduction is an essential pre-processing step in this application. PCA is a classic method for capturing variance and in this part, we will explore various PCA dimensions.

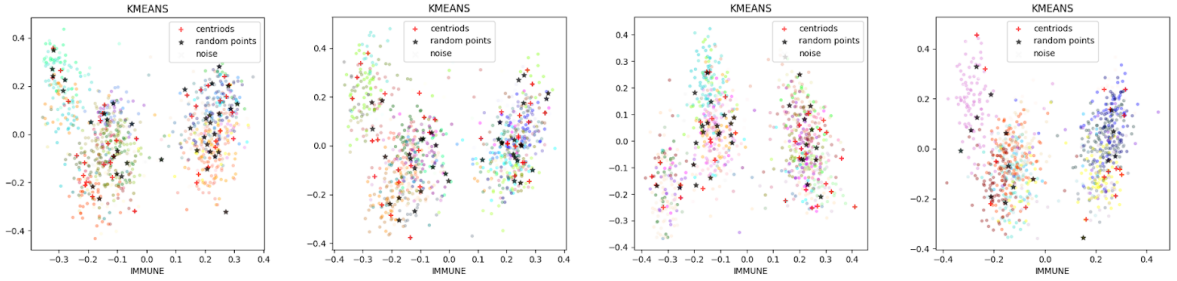


Dim = 2
Rouge Score: 0.0
Cosine Score(reduced): 1.1069
Cosine Score(full): 0.8067

Dim = 4
Rouge Score: 0.0044
Cosine Score(reduced): 1.0098
Cosine Score(full): 0.8260

Dim = 8
Rouge Score: 0.0111
Cosine Score(reduced): 1.1769
Cosine Score(full): 0.8834

Dim = 16
Rouge Score: 0.0010
Cosine Score(reduced): 1.0177
Cosine Score(full): 0.8270



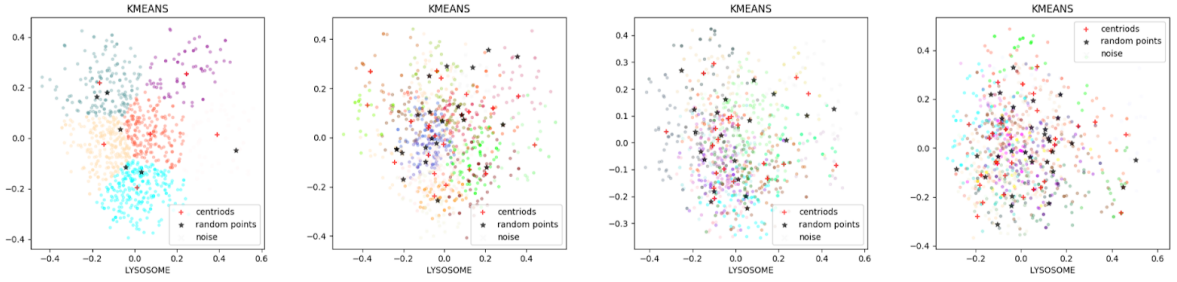
Dim = 32
Rouge Score: 0.0088
Cosine Score(reduced): 1.0123
Cosine Score(full): 0.8211

Dim = 64
Rouge Score: 0.0060
Cosine Score(reduced): 1.0068
Cosine Score(full): 0.8072

Dim = 128
Rouge Score: 0.0086
Cosine Score(reduced): 0.9927
Cosine Score(full): 0.7482

Dim = 256
Rouge Score: 0.0133
Cosine Score(reduced): 0.9787
Cosine Score(full): 0.6970

Figure 17. KMeans clusters of different PCA dimensions for 'PAUCITY'

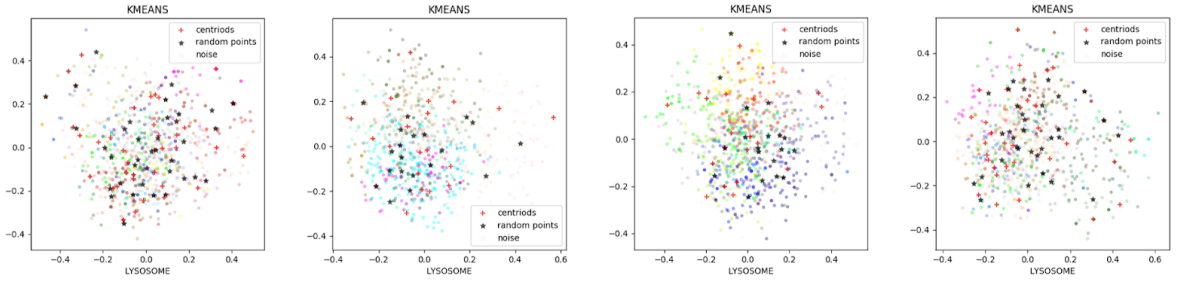


Dim = 2
Rouge Score: 0.0
Cosine Score(reduced): 1.1445
Cosine Score(full): 0.7877

Dim = 4
Rouge Score: 0.0096
Cosine Score(reduced): 1.0422
Cosine Score(full): 0.8227

Dim = 8
Rouge Score: 0.0083
Cosine Score(reduced): 1.0440
Cosine Score(full): 0.8268

Dim = 16
Rouge Score: 0.0085
Cosine Score(reduced): 1.0214
Cosine Score(full): 0.8273



Dim = 32
Rouge Score: 0.0089
Cosine Score(reduced): 1.0156
Cosine Score(full): 0.8293

Dim = 64
Rouge Score: 0.0128
Cosine Score(reduced): 1.0018
Cosine Score(full): 0.7991

Dim = 128
Rouge Score: 0.0106
Cosine Score(reduced): 1.0080
Cosine Score(full): 0.7663

Dim = 256
Rouge Score: 0.0172
Cosine Score(reduced): 0.9958
Cosine Score(full): 0.7527

Figure 18. KMeans clusters of different PCA dimensions for ‘LYSOSOME’

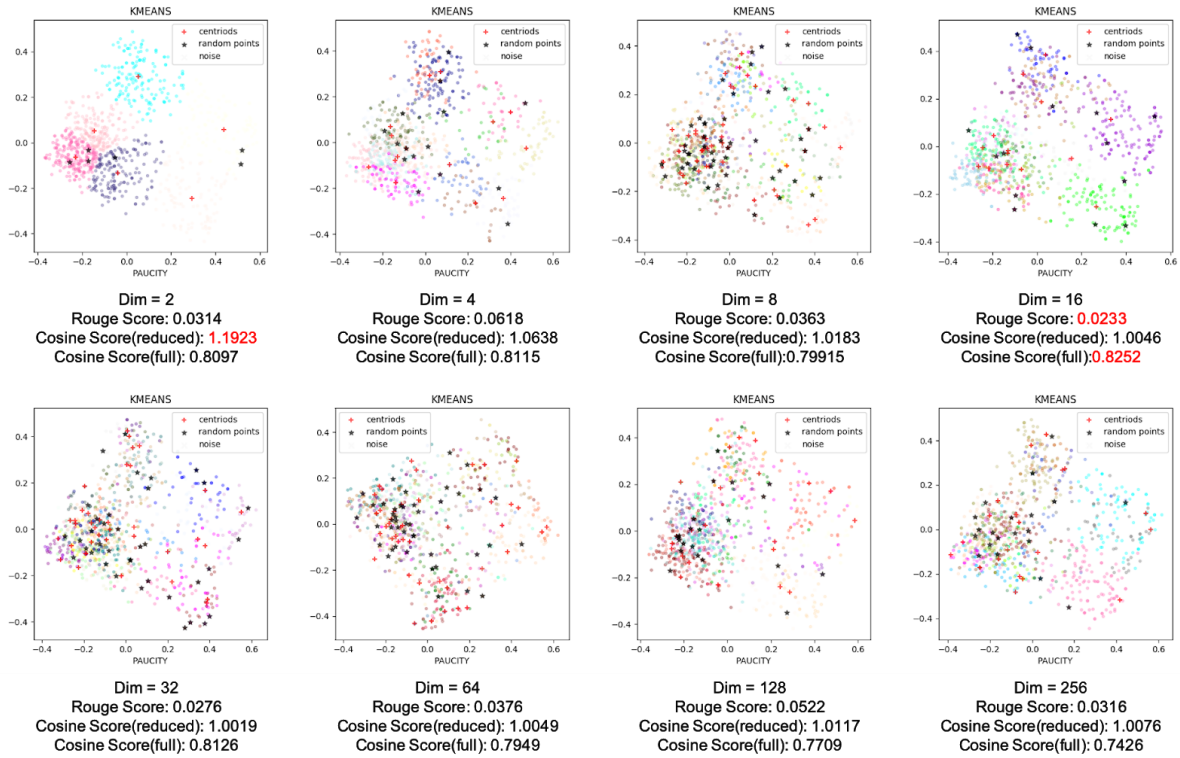


Figure 19. KMeans clusters of different PCA dimensions for 'IMMUNE'

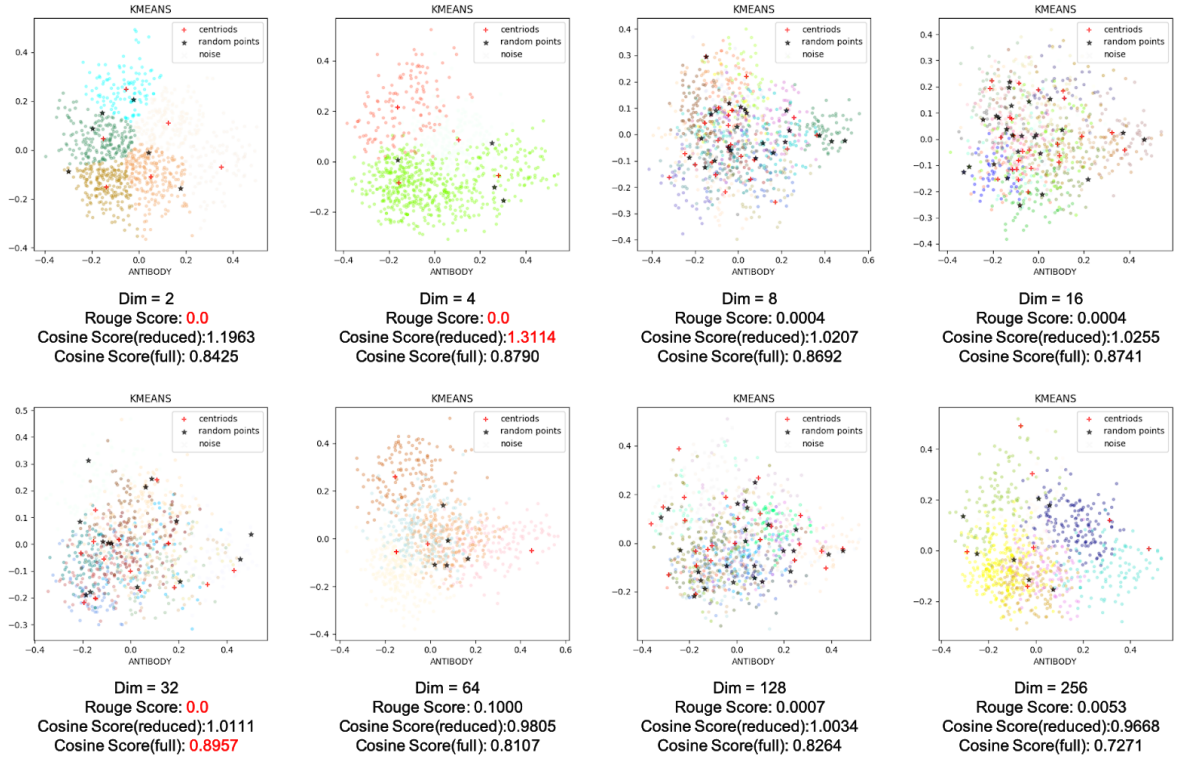


Figure 20. KMeans clusters of different PCA dimensions for 'ANTIBODY'

After investigating the PCA dimensions (figure 17. ~ figure 20.), there are several insights. Firstly, high dimension, (e.g., 64, 128, 256), deteriorates the clustering results in terms of rouge score and

cosine scores. As we mentioned in the Evaluation section, cosine scores have two versions, reduced version which is based on PCA dimension and full version which is based on full dimension of the features. Generally speaking, the reduced version cosine score was always higher than the full version. The expectation for cosine scores is the same that higher is better. But when the dimension was 64 or 128 or 256, both two kinds of cosine scores seemed to be lower. Secondly, dimension of 16 or 32 achieves considerable results and hence is a good choice for further experiments. Thirdly, as the dimension increases, the number of clusters raises as well. You might notice that some centroids were not centered in the cluster when PCA dimension was greater than 2. It is because we calculated them in a high dimension and then centroids were not central if projected into 2D space. We also discovered that the colors of some clusters were too light to display, like 'ANTIBODY' clustering results with a dimension of 4. There were only 2 obvious clusters but 4 centroids. The explanation was another two clusters were painted in a light color.

Table 5. Scores on different dimensions with fixed 10 clusters

PCA Dimension	Sum of PCA Explained Variance Ratio	Rouge Score(Kmeans)	Rouge Score(Random)	Cosine Score-Reduced (Kmeans)	Cosine Score-Reduced(Random)	Cosine Score-Full(Kmeans)	Cosine Score-Full(Random)
2	0.0613	0.0864	0.0959	1.0934	1.0000	0.8077	0.8233
4	0.1006	0.1068	0.0959	1.1017	1.0000	0.8198	0.8233
8	0.1624	0.0847	0.0959	1.0951	1.0000	0.8137	0.8233
16	0.2528	***0.0834***	0.0959	1.0641	1.0000	0.8260	0.8233
32	0.3736	0.0860	0.0959	1.0390	1.0000	***0.8303***	0.8233
64	0.5403	0.0954	0.0959	***1.0187***	1.0000	0.8294	0.8233
128	0.7219	0.1138	0.0959	1.0054	1.0000	0.7805	0.8233
256	0.8900	0.1497	0.0959	0.9781	1.0000	0.7035	0.8233

Table 5. gives the average rouge scores and cosine scores of eight query words, 'paucity', 'lysosome', 'compensatory', 'propagate', 'respiratory', 'immune', 'antibody', 'infection'. The number of clusters was fixed to 10 so for any score from random results is consistent. Like the ROUGE score of random picked instances was 0.0959. The Sum of PCA Explained Variance Ratio is how much variance captured by selecting this dimension. Rouge score is ROUGE-L in specific. And Cosine scores are calculated for features of reduced dimension and full dimension. Random scores mean the scores evaluated on randomly pick N instances where N equals to the number of clusters. The expectation was that the cluster results could beat the randomness. From the table 5., the minimum rouge score is 0.0834 when dimension equal to 16. This finding is consistent with clustering plots we have above. On the other hand, when dimension is over 64, cluster method hardly exceeds randomness in terms of Rouge evaluation.

Window Size

Window size is the number of tokens covered in the left/right context. Larger the window size is, more content would be included in the instances. This experiment focuses on discovering the

influence of different window sizes. In this experiment, we used SBERT vectorization and PCA dimension was 16.

Table 6. Scores on different window sizes

Window Size	Rouge Score(Kmeans)	Rouge Score(Random)	Cosine Score-Reduced (Kmeans)	Cosine Score-Reduced(Random)	Cosine Score-Full(Kmeans)	Cosine Score-Full(Random)
1	0.0	0.0301	**1.0825**	1.0000	0.4666	0.3882
3	0.0109	0.0072	1.0207	1.0000	0.7509	0.7499
5	0.0087	0.0092	1.0362	1.0000	0.8263	0.8175
10	**0.0046**	0.0056	1.0294	1.0000	0.8655	0.8505
15	0.0050	0.0066	1.0449	1.0000	**0.8691**	0.8584

This experiment is testing five different window sizes ranging from 1 to 15. As the length of instances is longer, the speed of processing will be slower and rouge score is likely to be smaller. However, there's no guarantee that bigger window sizes means better cluster results. Given Table 6, window size of 1 seems to be a good parameter.

Test results on 100 query words

In order to verify the validity of our method, we tested on around 100 query words including 50 high frequency words and 50 low frequency words. This experiment is based on KMeans method with PCA dimension of 16. The window size is fixed to 5.

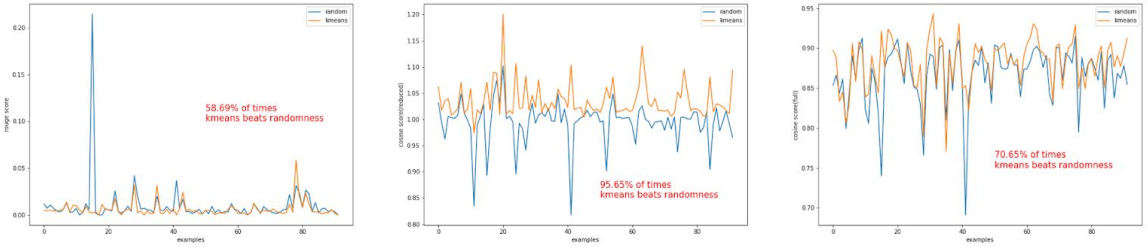


Figure 21. KMeans tests on 100 words from the covid corpus

Looking at figure 21, for around 58% of times, our model could achieve lower rouge scores than randomly selected instances. If looking at cosine scores in the reduced version, for over 95% of times clustering results were better than random results. But the percentage dropped to 70.65% for full version cosine score. It confirms that the PCA method captured the majority of variances and worked very well in this application.

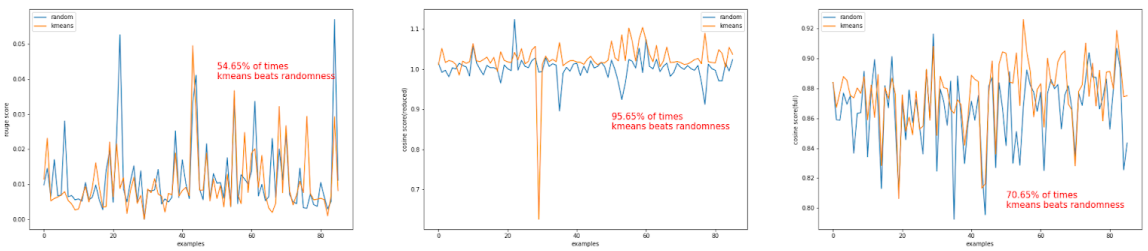


Figure 22. KMeans tests on 100 words from assnat corpus

Figure 22 is showing test scores on French words. In this case, for Rouge score, we got 54.65% of times that Rouge Score beat the randomness. And in terms of counting cosine scores, it turned out to be the same as we tested on English words, even though different line charts. In fact, hardly could I say ROUGE score was the accurate metric for evaluation because the baseline (random) scores varied due to different numbers of clusters. As we saw the cosine plots, it seemed 95% was a decent score to convince people of using KMeans. But as we mentioned in the Evaluation section, it was mainly because we applied the same strategy during clustering and evaluation phases. So personally, I would not say cosine was still a good evaluation metric.

Features and Distribution of Data Points' Neighbor Distances

Displayed below is the distance distributions of each data point were observed using different feature engineering methods with no dimension reduction. Sklearn's NearestNeighbors package was used to automatically derive these distributions when given the set of feature vectors retrieved with each feature engineering method. NearestNeighbors returns a distribution of distances between each data point and its closest neighboring data point. The query word here was 'set' and 500 instances were retrieved with a window size of 5. Epsilon is expressed on the y-axis of the graphs below, which represents Euclidean distance between two data points. This method can be used to determine the epsilon hyperparameter in DBSCAN, however it is shown that it is very sensitive, especially in the case of different feature engineering methods.

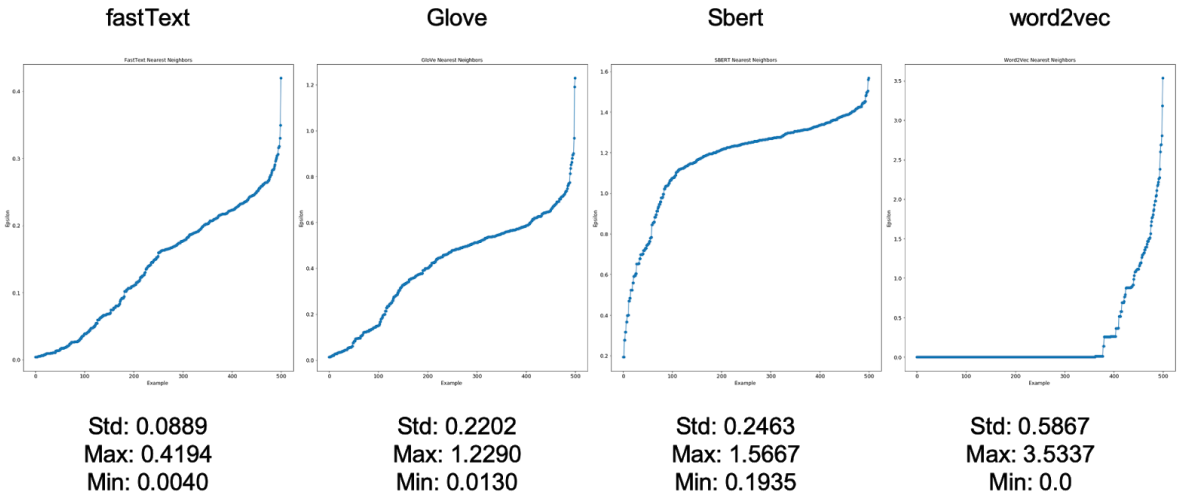


Figure 22. Distance distributions using different features

Here you can see that Word2Vec has the largest standard deviation and range between maximum and minimum distance values. Since Word2Vec was the only model trained on our corpus, this could be a possible reason why it most often produces the most distinct clustering distributions, or in this case the distance distribution, where in the example above we can see that many data points are essentially overlapping their nearest neighbor with the exception of many outlying, noisy data points.

FastText produced a very small range of distances, but the trend is almost linear; which is very similar to trend in the distance distribution for GloVe. The closer the trend of the graph is to being linear, the more possible it is that the data points form a blob, in which points expand from mostly a single epicenter; meaning that the distribution of points does not necessarily have multiple distinguishable clusters.

SBERT seems to have few neighboring points that are close to the minimum distance found in its distribution of distances. However, there is a somewhat horizontal trend in the distribution when plotted, just that this trend is closer to its maximum distance. This implies that there are less distant outliers, and that the data points generated with SBERT are in general mostly further apart than those generated with other methods. Since no dimension reduction is used and SBERT has the highest amount of dimension in its features, this is likely a contributing factor to the higher frequency of higher magnitudes of distance.

Clustering on Different Features

When further analyzing different feature engineering methods available in feature_extraction, we used KMeans. PCA was used to reduce features down to a dimension size of 16, inspired by previous results. For each word tested, a window size of 10 was utilized with up to 1000 data points (instances).

Table 7. Results for query word ‘PAUCITY’

Feature Types	PAUCITY	Number of Clusters	Rouge Score
Fasttext	- has a variety of limitations . Apart from the PAUCITY of demographic data , self reporting questionnaires are largely limited animal species . Moreover , there has been a relative PAUCITY of therapeutic candidates that have translated from animal models into event data . Logistically this epidemic model and the relative PAUCITY of information and research on medical errors lend themselves to such as hereditary , and the conditions in biliary and PAUCITY the bile , also with changes the resembling infectious .	41	0.0190
Glove	large number of animal studies , there is a relative PAUCITY of clinical data . There have been a number of . With an apparently high frequency of infection but relative PAUCITY of information pertaining to the impact of CoV in patients region provided below . Overall , there is a general PAUCITY of reports in which H. influenzae strains and antibiotic sensitivity acids in the brain of patients with schizophrenia revealed a PAUCITY of virus . We hypothesized that T cells found in exacerbation of IPF [4] . Second , the PAUCITY of sampling data from BALF is another limitation . However	31	0.0200
Sbert	concentrated in Beijing , particularly at Tsinghua University . ostensible PAUCITY of research activity using cryo-ET in China . These deficiencies [2] [3] [4] The PAUCITY of relevant evidence in the preschool age group might explain probe direct interactions with remorin proteins . There is a PAUCITY of tools to study vector biology and vectorvirus-host interactions at , at least in part , be attributed to the PAUCITY of animal models that manifest insulin resistance while replicating adequately as a form of tourist travel . There is a PAUCITY of studies which investigate other forms of public transport such , the panel considered the high case fatality and the PAUCITY of possible pharmacological alternatives in all of the recommendations .	31	0.0167
Word2vec	nontoxic) . Perhaps the most serious concern is the PAUCITY replicate the pathogenesis of human disease , which of animal at forecasting through a very engaged modeling community , the PAUCITY of case transmission data was cited as one of the This gap may be explained in part by the current PAUCITY of scientific advances in genomics that have practical applications to work on] vaccine development OMP , B1 vaccine is PAUCITY chemically LBP on , the (the PT iron-repressed haemagglutinin	9	0.0182

Table 8. Results for query word ‘IMMUNE’

Feature Types	IMMUNE	Number of Clusters	Rouge Score
Fasttext	rGST-COE and rGST-S1D also showed significant reactions with pig anti-PEDV IMMUNE sera (Fig. 6) . To test whether immunization The specific IgY opsonized the bacterium and intensified the cellular IMMUNE response , probably due to the recognition of IgY by 1976) showed that genetically resistant mice had an altered IMMUNE response when inoculated with MHV-3 , even though no apparent as crystal contacts . BHP includes mostly interaction with human IMMUNE response proteins . BSL is derived from small ligands seen and suggest that activation of apoptosis pathways via the innate IMMUNE pathway may contribute to the tolerance of henipaviruses by flying present clearance in aeruginosa bacteria contributing identified by recognition . IMMUNE are polarized (inflammatory phenotype cytokines thereby , from removal	24	0.0063
Glove	factors , perhaps genetically determined , that regulate both the IMMUNE response to infections and the risk of developing allergies or immune response in the two host types : Mutations facilitating IMMUNE escape or tolerance in the first host might cause the (Flagellin-SAPN) . This demonstrated the induction of cellular IMMUNE responses and generation of a memory pool by the vaccine and vascular cell functions , sex steroids also modulate adaptive IMMUNE and inflammatory functions . My summary of these complex and	9	0.0075
Sbert	fundamental differences in how commensal microbes control activation of unique IMMUNE components that protect against infection with viral rather than fungal affects 25 % of general populationAllergic : IgE-mediated release of IMMUNE mediators (histamine , leukotrienes , etc.) from mast Ad-UMAS) is a potential vaccination route to elicit protective IMMUNE responses . Compared to other immunization routes , oral and through driven levels single-gene Mendelian to , 25,000 then , IMMUNE that system we every area annotation also search for recent with subsequent destruction being caused by a combination of host IMMUNE responses and dermonecrolytic fungal toxins . 5 Various virulence factors	30	0.0052
Word2vec	the number of susceptible , infected and recovered (and IMMUNE) individuals . R0 can be calculated by inferring the or through xenotransplantation of hematopoietic stem cells (creating human IMMUNE system mice , known as HIS) and/or other human affected animals because they can alter the physiology (notably IMMUNE function) of experimental subjects or experimental endpoints such as and viral E3 protein) , thus abrogating the antiviral IMMUNE responses Lindner et al. , 2005 ; Loo et al. al. 2014) . The suppression of the host antiviral IMMUNE response to facilitate higher virus production during ADE of DENV	8	0.0

As we can see in both tables 7 and 8, GloVe performs last in terms of ROUGE score, while Word2Vec and SBERT compete for first place. It's not clear whether comparing feature extraction methods based on ROUGE scores alone are appropriate. Some feature extraction methods may represent different aspects of the inputs differently based on semantics, syntax, etc.. Therefore, it's helpful to also analyze some concordance results and compare.

The syntax and semantics diversity of the concordances can differ between feature extraction, and even randomness that is from the KMeans clustering algorithm which can affect the number of 'K' clusters, demonstrating that the choice of features used may be dependent on preference. As seen before, the larger window size can also affect the results, which can make it a bit more difficult for linguistic comparison.

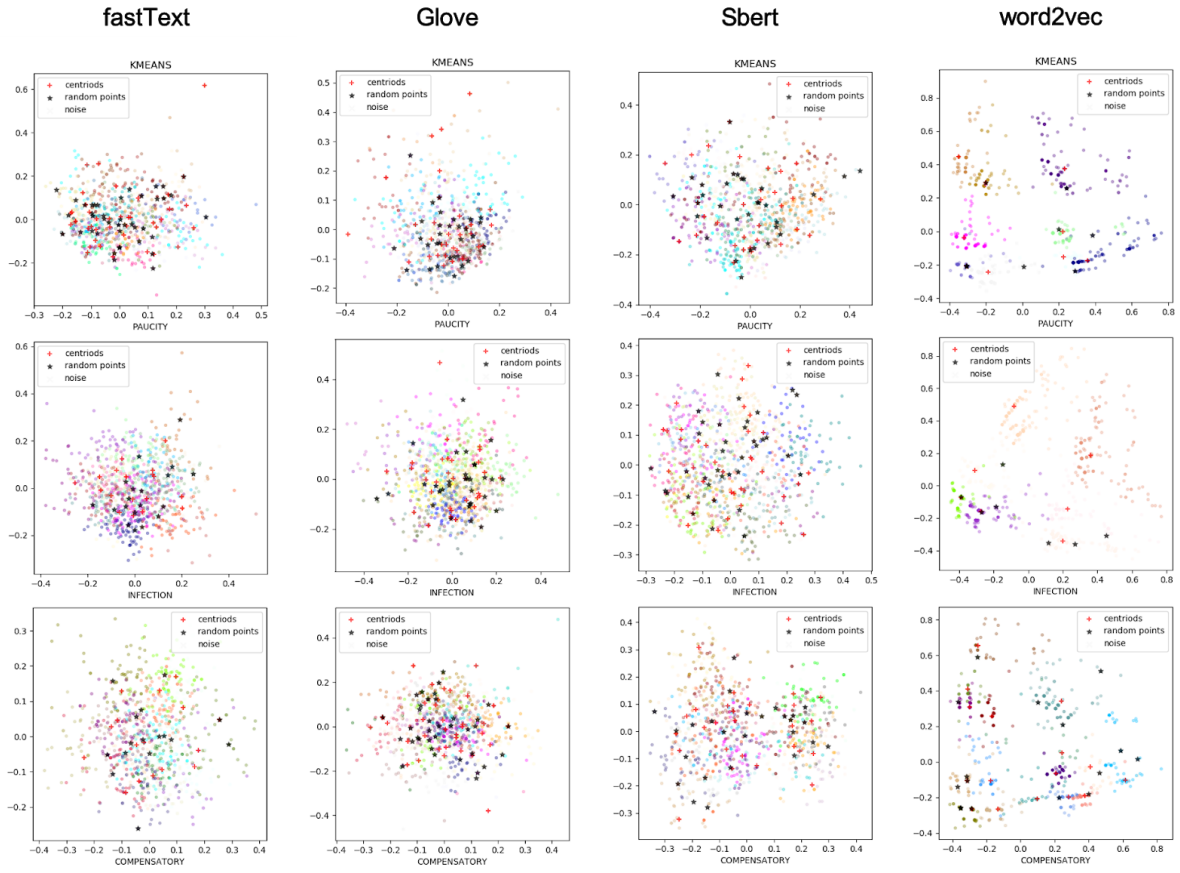


Figure 23. Cluster results by using different features

In most cases it seems that the data points generated with Word2Vec form more easily distinguishable clusters in two dimensional visualizations. There are multiple occasions where some feature extraction methods produce distributions of data points that have no obvious cluster distinctions to the human eye, when reducing to two dimensions for visualization; which is partially due to PCA not being able to optimally preserve the variance when reducing very high dimensional vectors to two dimensions.

Fundamental Differences in Application of Feature Engineering Methods

GloVe and Word2Vec derive word embeddings on a token level, FastText derives embeddings on a character level, whereas SBERT derives sentence embeddings on a sentence level. While in application, all methods will result in left and right context representations, these fundamental differences can have larger ramifications, especially since SBERT is

pretrained to encode complete sentences but is most often encoding partial sentences in our application. This is because the left and right contexts are dependent on window_size, a parameter of our elasticsearch query, which determines how many tokens will be in each context. Because of the difference in pre-training methods and our applications input for SBERT, this might have potential performance consequences in our concordancer.

Additionally, due to limitations of computational resources and time, Word2Vec is the only model trained on the actual corpus while the other methods used pre-trained embeddings. Meaning that although GloVe and Word2Vec both encode tokens into vector representations, out of vocabulary (OOV) issues may be less significant in the case of Word2Vec. This is especially because the COVID 19 corpus is a very unique domain that may contain many OOV tokens that were not in the pretrained GloVe model. Ideally, we would want to avoid as many OOV tokens as possible, because otherwise we would be including many average vectors derived from the model's vocabulary, which will inevitably move all data points closer in hyperdimensional space. This is likely part of the reason why results from using KMeans with PCA dimension reduction on Word2Vec features seem to be the most successful and desirable because these data points form the most distinguishable clusters.

For evaluating performance and behavior of feature engineering methods, ROUGE score seems to be fundamentally limited for this task. While the Word2Vec model constantly had better ROUGE scores, this could be more relevant to the model producing better data points for clustering. If all of our feature engineering methods were optimized through proper training and/or fine tuning onto our corpora, ROUGE may not capture the real differences. For instance, SBERT may extract more semantically oriented features, while our Word2Vec model may be extracting more syntactically oriented features. It is unknown if ROUGE may be biased towards one way or another in terms of syntactically or semantically different sequences.

Future work

Based on the challenges we identified during the implementation of this project, we came up with the following list of possible directions and methods worth exploring:

- Since ROUGE is a n-gram overlap based approach, it doesn't take into account the semantic level information contained in the vector representations of the samples in a concordance output. There are some other ROUGE variants worth exploring like ROUGE - synonyms and ROUGE - AR, we couldn't test these variants due to challenges associated with the type and size of the data.
- Weighting words based on index in the sequence could change the effect of window size on clustering significantly (likely mitigating/reducing the correlation with high window size equating to high amount of clusters)
- Creating our own models for both token-based and context-based vectorization, by fine tuning GloVe or further training Word2Vec and by training our own SBERT models from scratch to take incomplete sentences and be better tuned for our corpora domains

- Defining desired linguistic differentiation between concordances to better optimize for desired results

Timeline

Week No.	**Deliveriables**	**Nicholas**	**Komal**	**Shonna**
1. May 4 ~ May 10	- Team Contract - Project Plan - Prepare Data and Reading Data - Test out nltk.concordancer() on slime files	- Project Plan(Methods Section) - Complete Mural pdf	- Project Plan(Description & Expected deliverables Sections) - Teamwork Contract	- Project Plan(Datasets & Schedual Sections) - Prepare Data(code)
2. May 11 ~ May 17	- Use Elasticsearch to return instances that contain a target word - Try sklearn k-means to quickly build a prototype - Build a baseline based on nltk.concordancer()	- Sklearn Clustering (Minibatchkmeans) - Picking elbow and make decisions	- A baseline on Nltk.concordancer() - Use ROUGE to evaluate the score	- Build a corpus reader to extract files from zipfile - Use Elasticsearch to retrieve instances that contain a target word
3. May 18 ~ May 24	- Implement other clustering algorithms: DBSCAN - Add another features: SBERT - Integrate search, extract feature, clustering and evaluation	- Solve the problem of picking an optimal K - Implement DBSCAN	- Use ROUGE to evaluate the score	- Solve the problem of retrieving instances from huge datasets by filtering documents first and avoiding nested search in Elasticsearch - Integrate SBERT as the sentence feature
4. May 25 ~ May 31	- Integrate source codes - Add another cluster algorithm: OPTICS - Add another feature: Fasttext - Test performance	- Integration of clustering and Rouge score - Implement OPTICS - Fix word2vec - Fix bugs and add asserts	- Fasttext - Research medical wordnet	- Integration of elasticsearch and clustering - Fix bugs when searching - Add 2 methods to reduce dimensions: PCA, TSNE

				- Fix bugs and add asserts
5. June 1 ~ June 7	- Test on French Corpus - Add Glove feature - Experiments on picking hyper-parameters of cluster algorithms	- Successfully get glove embeddings based on our own corpus - Experiments on picking eps - Get statistics of our corpus - Fix Rouge score	- Fasttext is done - Fix Rouge score	- OPTICS: use core_distance to get centroids - Reorganize the elasticsearch to reduce searching time - Manage full pipeline and support comparison with randomly picked instances
6. June 8 ~ June 14	- Switch to Regular Kmeans - Experiments on features, window size, dimensionality reduction -Final Report	- Experiments on comparison between features - Final report (analysis section, glove, and kmeans)	- Final report (Introduction, Method, Evaluation and Future word sections)	-Visualizations - Tests on dimensionality reduction, window size - Final report(Data, analysis section)
7. June 15 ~ June 19	- Python package - Final Report	-Visualization - Final report(proofreading, analysis)	-Python package	-Visualization - Final report(analysis, timeline)
8. Jun 22 ~ Jun 30	- Final Report - Data Product - Video Presentation			

References

1. Luhn, H.P. (1960), Key word-in-context index for technical literature (kwic index). Amer. Doc., 11: 288-295. doi:10.1002/asi.5090110403

2. George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Proceedings of the workshop on Human Language Technology* (*HLT '93*). Association for Computational Linguistics, USA, 303–308. DOI:https://doi.org/10.3115/1075671.1075742

3. Fatih Yavuz,The Use of Concordancing Programs in ELT, Procedia - Social and Behavioral Sciences,Volume 116,2014,Pages 2312-2315, ISSN 1877-0428, https://doi.org/10.1016/j.sbspro.2014.01.565.

4. Elastic search documentation
   Elasticsearch Reference [7.7]

5. SBERT (Generating semantically meaningful sentence embeddings)
   Sentence level embeddings from BERT | dair.ai
   Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks

6. fastText (FastText under the hood)
   FastText: Under the Hood
   [1607.04606] Enriching Word Vectors with Subword Information

7. KMeans inertia plotting
   Tutorial: How to determine the optimal number of clusters for k-means clustering

8. PCA
   Principal Component Analysis for Dimensionality Reduction

9 T-sne
   PCA vs t-SNE: which one should you use for visualization

10. Optics clustering
    Clustering Using OPTICS

11. DBSCAN Clustering
    How DBSCAN works and why should we use it?

12. ROUGE
    What is ROUGE and how it works for evaluation of summarization tasks? | RxNLP

13. GloVe
    GloVe: Global Vectors for Word Representatiosn

14. Word2Vec
    Distributed Representations of Words and Phrases and their Compositionality