Natural Language Processing (NLP)-Based Text Clustering

Agenda

01 Problem Definition

02 Data Understanding

03 Data Cleaning

04 Text Visualization

05 Model Building

06 Conclusion

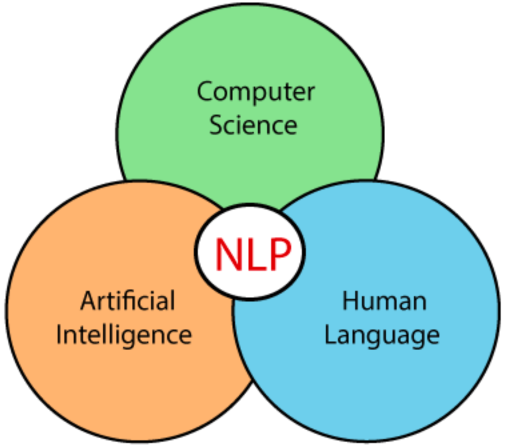
Problem Definition

- 1180 sentences pertaining to different religions without labels are given to us.
- Explore different methods of optimal no. of clusters
- Experiment Depth Difference (DeD) method
- Use embedding techniques to convert text data into numeric

Objective

- We need to identify optimal number of clusters and label them accordingly.

Why NLP?



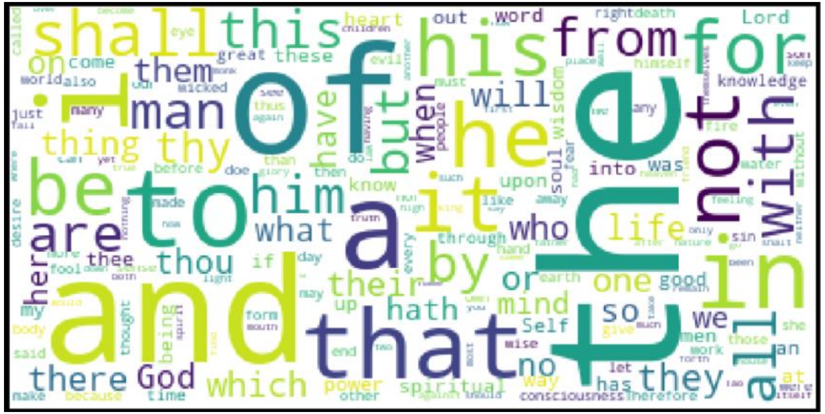
- Ability of a computer to understand, analyze, manipulate, and potentially generate human language.
- Real Life Applications of NLP
  - Text Summarization
  - Question Answering
  - Spam detection
  - Sentiment Analysis
  - Machine Translation – from one language to another

Data Understanding

```
file = open("Complete_data.txt", 'r')
lines = file.readlines()
print("No. of lines in data : ", len(lines))
```

```
No. of lines in data :  1180
```

```
['0.1\n',
 '§ 1. The Buddha: "What do you think, Rahula: What is a mirror for?"The Buddha:Rahula: "For reflecti
on, sir."Rahula:The Buddha: "In the same way, Rahula, bodily acts, verbal acts, & mental acts are to
be done with repeated reflection.The Buddha:"Whenever you want to perform a bodily act, you should re
flect on it: \'This bodily act I want to perform — would it lead to self-affliction, to the afflictio
n of others, or to both? Is it an unskillful bodily act, with painful consequences, painful results?
\' If, on reflection, you know that it would lead to self-affliction, to the affliction of others, or
to both; it would be an unskillful bodily act with painful consequences, painful results, then any bo
dily act of that sort is absolutely unfit for you to do. But if on reflection you know that it would
```



Many common words like the, and, that, a etc. are present in the given data.

Data Cleaning & Pre-processing

Cleaning

Remove Punctuations, special characters, numbers etc.

LowerCase

Convert clean text to lower case

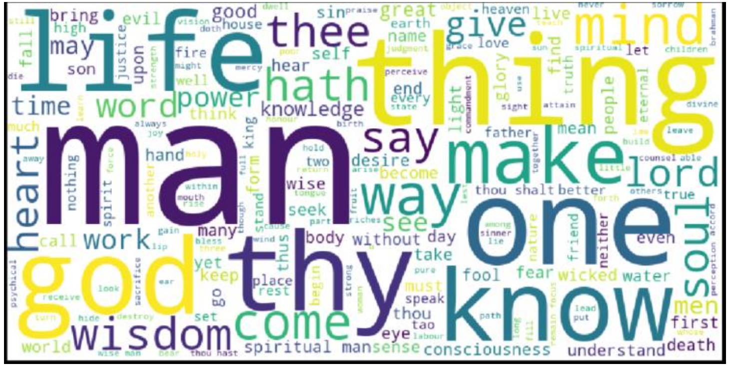
Tokenization

Split long text into individual words

Lemmatization

Convert words to Base form

Clean lines : 589




```
0      buddha think rahula mirror buddha rahula refle...
1      bless one stay kosambi simsapa tree grove pick...
2      stress know cause stress come play know divers...
3      vision arise clear know arise discernment aris...
4      sariputta three form stressfulness friend stre...
                              ...
584    condemn makers worshippers idols thou god art ...
585    worthily punish destroy multitude beasts inste...
586    thy judgments lord great thy word express ther...
587    intercession sedition occasion core thy saint ...
588    creatures obey god order service good punishme...
Name: sentences, Length: 589, dtype: object
```

Text Embedding (TF-IDF)

- TF-IDF transformed corpus of 589 lines into 589 x 6102 sparse matrix

```
# Converting text data to numeric using TFIDF word embedding technique
vectorizer = TfidfVectorizer(stop_words='english')
vectors  = vectorizer.fit_transform(corpus)
feature_names = vectorizer.get_feature_names()
dense = vectors.todense()
denselist = dense.tolist()
tf_df = pd.DataFrame(denselist, columns=feature_names)
```

```
tf_df.shape
```

```
(589, 6102)
```

```
tf_df
```

	aaron	abandon	abase	abasement	abash	abate	abateth	aberrant	abhor	abhorreth	abide	abideth	abilities	ability	abiron	able	abnormal	abo
0	0.0	0.00000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	
1	0.0	0.00000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	
2	0.0	0.00000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	
3	0.0	0.08468	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	
4	0.0	0.00000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
584	0.0	0.00000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	
585	0.0	0.00000	0.0	0.0	0.0	0.0	0.052032	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.028090	0.0	
586	0.0	0.00000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.041435	0.0	

Text Embedding (Count Vectorizer)

- Count Vectorizer transformed corpus of 589 lines into 589 x 6259 sparse matrix

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
cv_vector=cv.fit_transform(corpus).toarray()
cv_df = pd.DataFrame(cv_vector, columns = cv.vocabulary_)
cv_df.shape
```

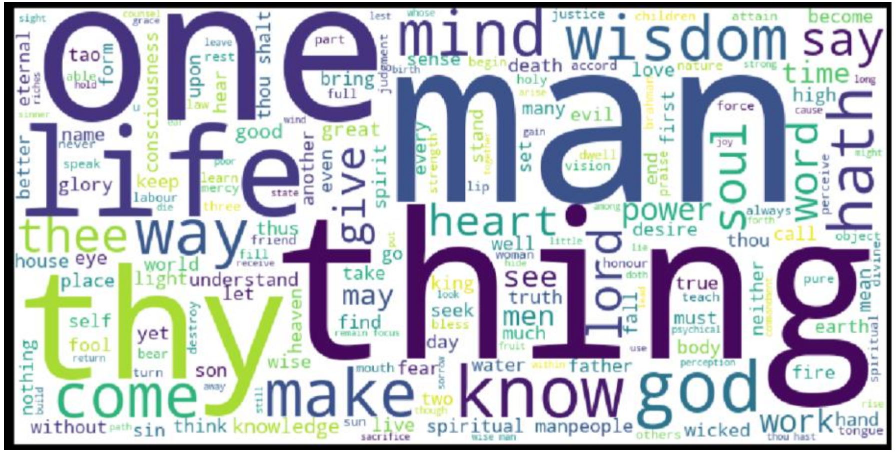
```
(589, 6259)
```

```
cv_df
```

	buddha	think	rahula	mirror	reflection	sir	way	bodily	act	verbal	mental	do	repeat	whenever	want	perform	reflect	would	lead	self	affliction
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Dimensionality Reduction

- 589 clean lines with 6102 different words present in given data
- Vector matrix after embedding is Sparse with many zeros
- PCA helps transform the matrix into smaller dimensions



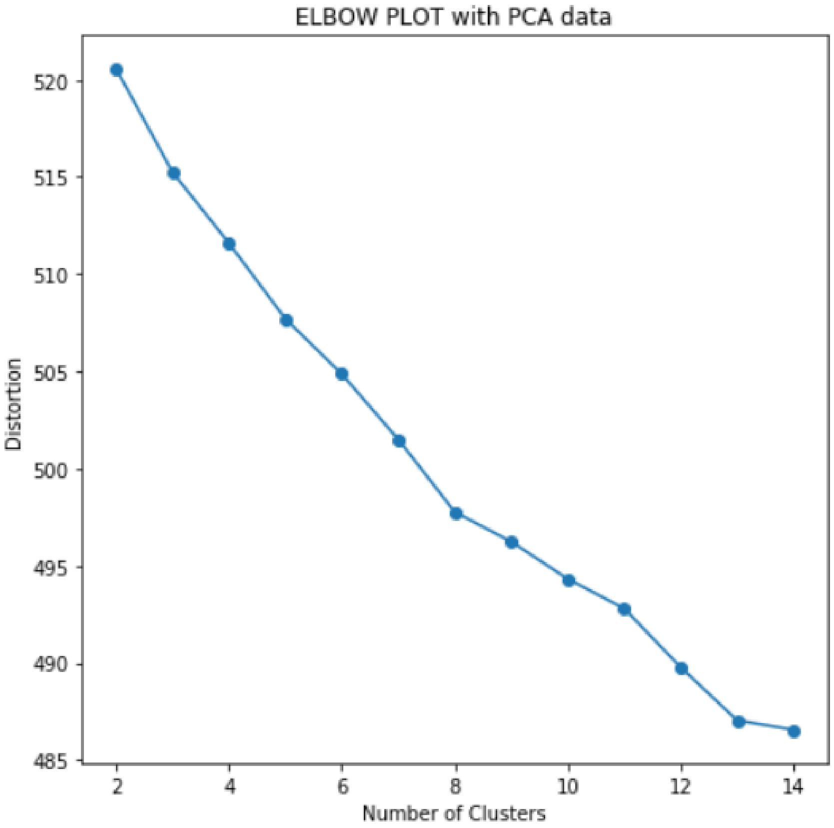

```
#Dimentionality reduction using sklearn PCA
pca = PCA(n_components=0.95)
tf_pca = pca.fit_transform(tf_df)
tf_pca.shape
```

```
(589, 487)
```

```
tf_pca
```

```
array([[-0.11175122, -0.05027314, -0.03121333, ...,  0.02727655,
         0.02198972,  0.06015316],
       [-0.09376615, -0.05293873, -0.04079513, ..., -0.05943734,
        -0.03208265, -0.02229695],
       [-0.10998769, -0.08626706, -0.06310916, ..., -0.00048585,
         0.02653828,  0.03310622],
       ...,
       [ 0.03561542, -0.04031887,  0.00417432, ...,  0.01390312,
         0.00712102, -0.01092679],
```

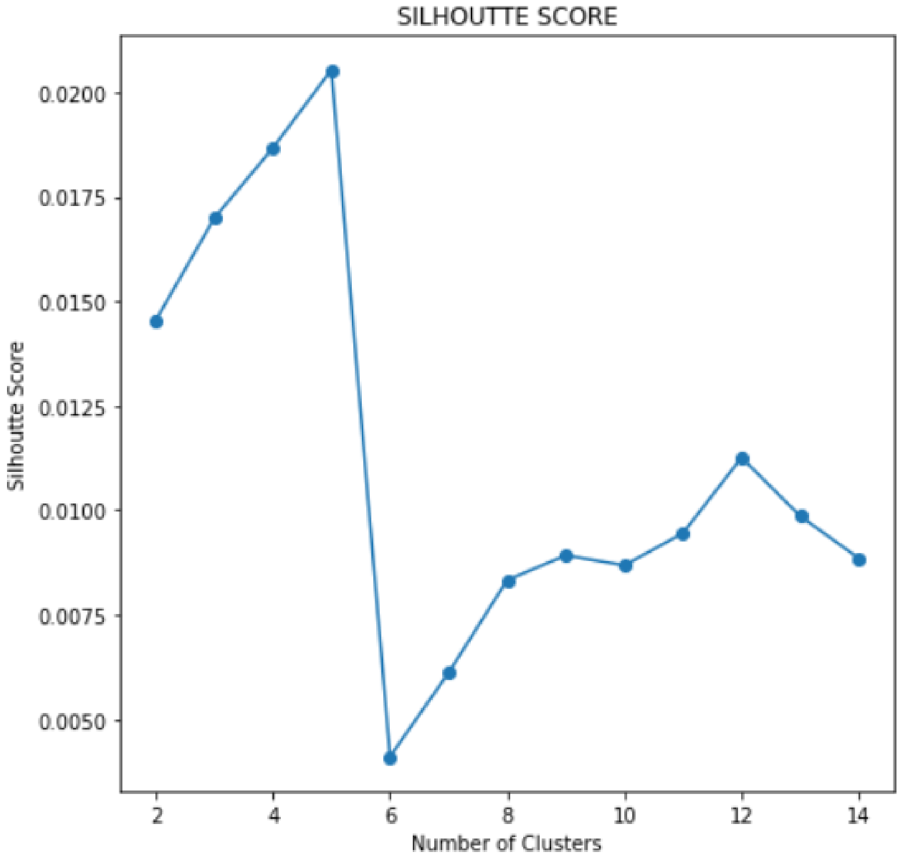
Find Optimal K – Elbow Method



In the Elbow Method, we pick a range of candidate values of k, then apply K-Means clustering using each of the values of k.
We then find the average distance of each point in a cluster to its centroid, and represent it in a plot.

- No clear Elbow, though slight bend at K=8 indicating reduction in WCSS.

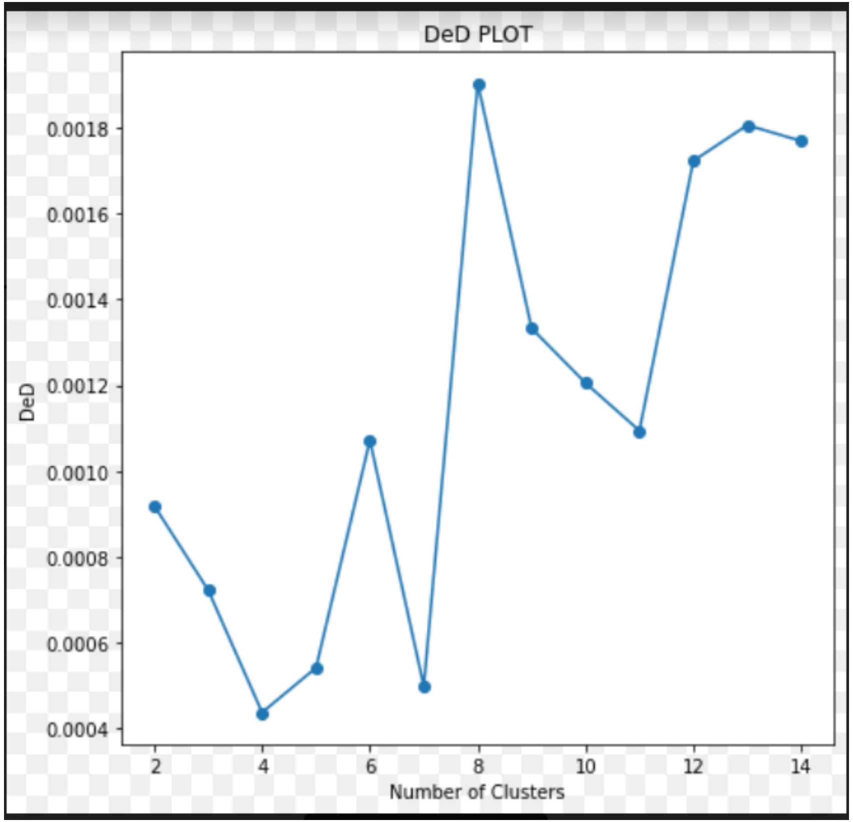
Find Optimal K – Silhouette Method



Silhouette Coefficient or silhouette score is a metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1.
1: clusters are well apart from each other and clearly distinguished.
0: clusters are indifferent, or we can say that the distance between clusters is not significant.
-1: clusters are assigned in the wrong way.

- Silhouette Method shows optimal score for K=5

Find Optimal K – Depth Difference Method



DeD method is used to estimate the optimal cluster based on the data depth.
It estimates the k before the actual clustering is constructed.
Data depth assigns the value between 0 to 1 to each data point which specifies centrality / deepness of a point in the data set.
Point with max depth will be the deepest point in the data set.

- Depth Difference method estimated optimal K value as 8

Clustering Models – K Means

	Sentence	Cluster
0	buddha think rahula mirror buddha rahula refle...	5
1	bless one stay kosambi simsapa tree grove pick...	3
2	stress know cause stress come play know divers...	3
3	vision arise clear know arise discernment aris...	3
4	sariputta three form stressfulness friend stre...	5
...	...	...
584	condemn makers worshippers idols thou god art ...	7
585	worthily punish destroy multitude beasts inste...	2
586	thy judgments lord great thy word express ther...	5
587	intercession sedition occasion core thy saint ...	2
588	creatures obey god order service good punishme...	2

Output of Kmeans model built with K=8

Cluster 0: Words like act, teach, soul, mind, life come together in Book of Wisdom
Cluster 1: Words like psychic, spiritual, Conciousness, perception come together Yoga Sutra book
Cluster 2: Words like Tao, Sage, Heaven come tohether in Tao Te Ching spiritual book
Cluster 3: Words like Teacher, nature, heart, object come together in Ecclesiastes
Cluster 4: Words like Nachitekta, death, Yama, Vayu, mortal come from Upanishad
Cluster 5: Words like Monk, Cessation, discernment come together in Buddhisam book
Cluster 6: Words like thee, thy, god, evil, lord come together in book of proverb
Cluster 7: Words like wicked, fool, evil are from book of Ecclecisiasticus

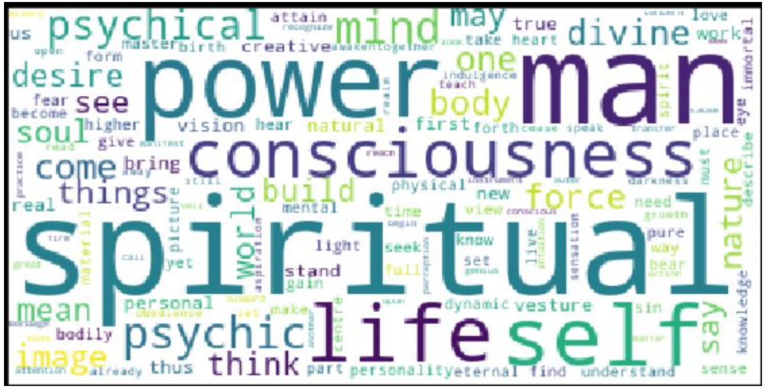
Clusters numbers assigned logical Name based on books contents & WordCloud of each cluster

WordCloud of Resultant Clusters

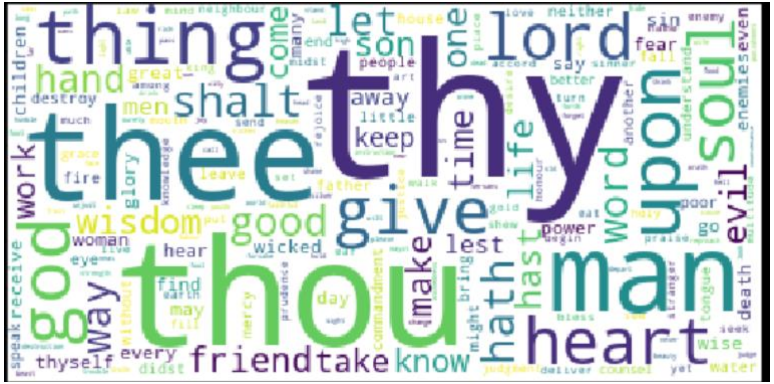
Tao Te Ching

Upanishad

Yoga Sutra

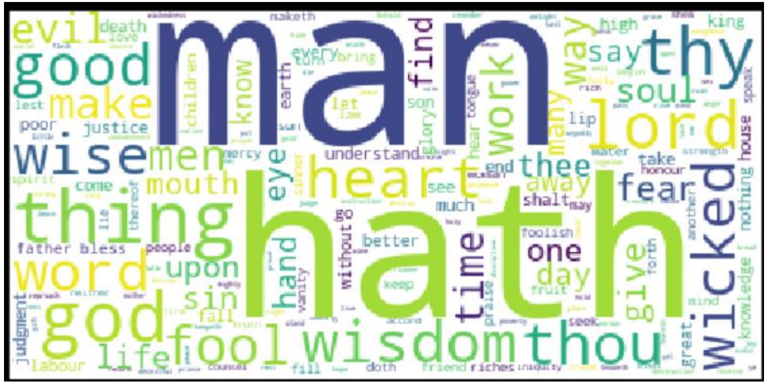


Book of Proverb

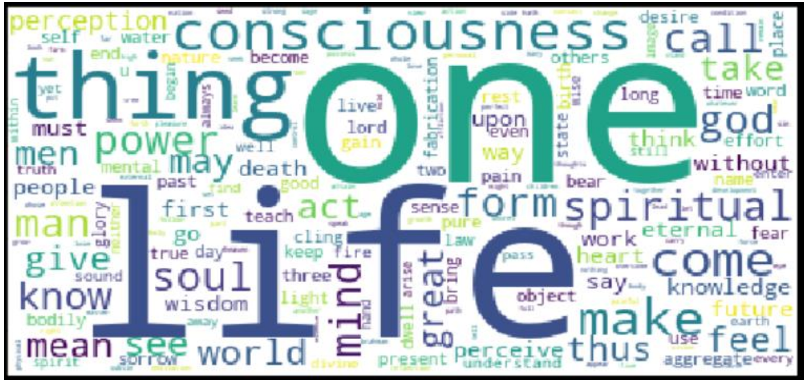


WordCloud of KMeans Clusters

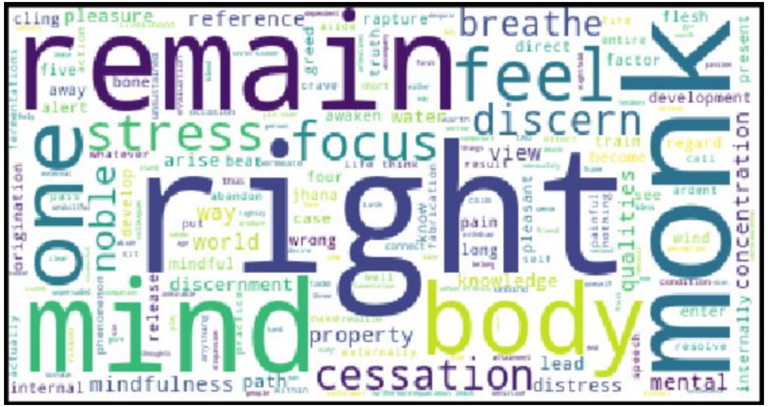
Book of Ecclesiasticus



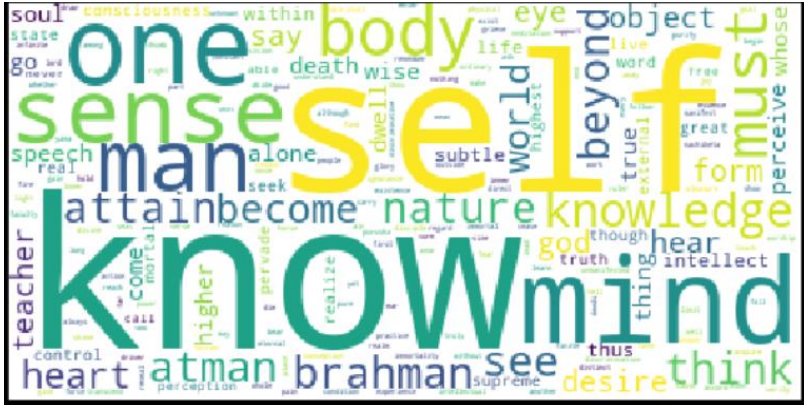
Book of Wisdom



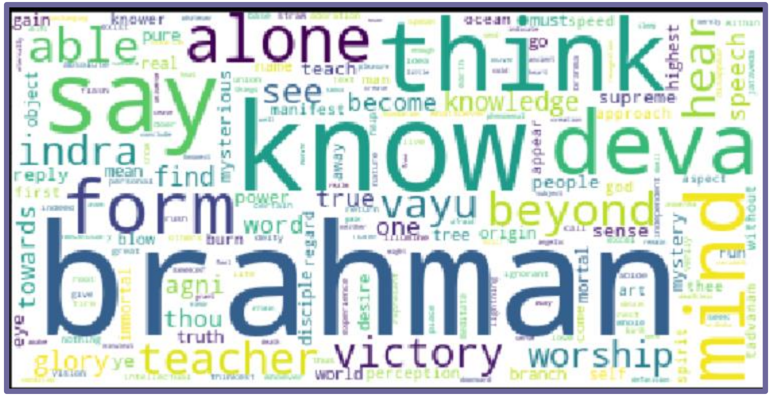
Buddhism

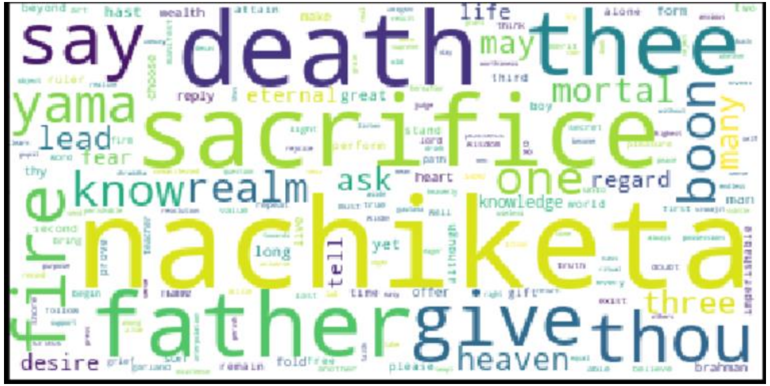


Book of Ecclesiastes



Incorrect Results with K=7





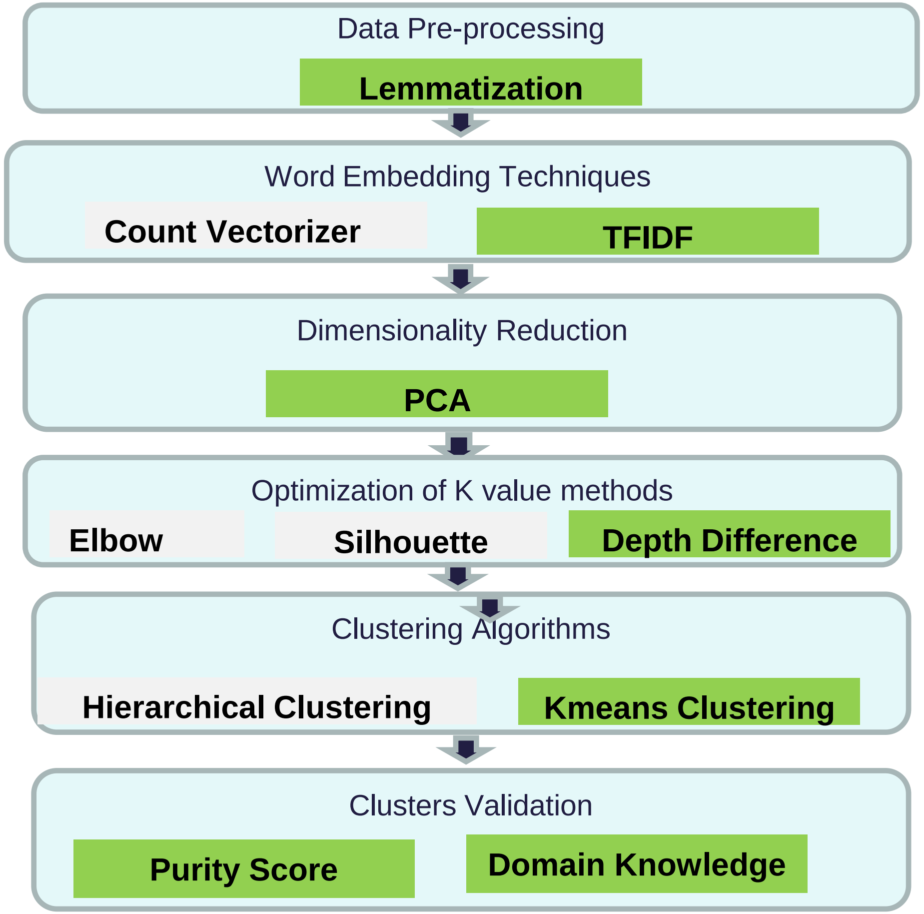
- Words in both the clusters represent Upanishad
- Words like Brahman, Vayu, Indra, deva are from Upanishad, but incorrectly classified into separate cluster.

Purity Scores Comparison

- Comparison of purity scores across models
- All below models were run with **K=8**

				Purity Score
Model 1	Count Vectorizer →	PCA →	KMeans	**0.41**
Model 2	TFIDF Vectorizer →	PCA →	KMeans	**0.67**
Model 3	TFIDF Vectorizer →	PCA →	Hierarchical clustering	**0.63**

Conclusion



- We were able to cluster the given text into 8 logical names representing different spiritual books:

    1. Book of Wisdom
    2. Yoga Sutra
    3. Tao Te Ching
    4. Book of Ecclesiastes
    5. Upanishad
    6. Buddhism
    7. Book of proverb
    8. Book of Ecclesiasticus

- **Kmeans** turned out to be the best model, with :
    - Lemmatization
    - TF-IDF
    - PCA applied
    - And **K =8**

- **Main challenges** were :

    - **Finding Optimal K** value
    - **Validating clusters** applying domain knowledge

Thank You

Appendix

References

- https://link.springer.com/article/10.1007/s41019-019-0091-y
- https://devopedia.org/text-clustering
- https://towardsdatascience.com/natural-language-processing-nlp-for-machine-learning-d44498845d5b
- https://medium.com/mlearning-ai/nlp-tokenization-stemming-lemmatization-and-part-of-speech-tagging-9088ac068768
- http://brandonrose.org/clustering
- https://towardsdatascience.com/a-friendly-introduction-to-text-clustering-fa996bcefd04
- https://www.datacamp.com/community/tutorials/stemming-lemmatization-python
- https://blog.bitext.com/what-is-the-difference-between-stemming-and-lemmatization/
- https://towardsdatascience.com/word-embeddings-for-nlp-5b72991e01d4?gi=552c2704f14
- https://www.geeksforgeeks.org/word-embeddings-in-nlp/
- https://medium.com/@b.terryjack/nlp-everything-about-word-embeddings-9ea21f51ccfe
- https://towardsdatascience.com/understanding-word-embeddings-with-tf-idf-and-glove-8acb63892032
- https://medium.com/@joshsungasong/natural-language-processing-count-vectorization-and-term-frequency-inverse-document-frequency-49d2156552c1
- https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa
- https://towardsdatascience.com/silhouette-method-better-than-elbow-method-to-find-optimal-clusters-378d62ff6891
- https://en.wikipedia.org/wiki/Cluster_analysis#External_evaluation
- https://sanjayasubedi.com.np/nlp/nlp-with-python-document-clustering/

01 NLP Libraries

Scikit-learn: It provides a wide range of algorithms for building machine learning models in Python.
Natural language Toolkit (NLTK): NLTK is a complete toolkit for all NLP techniques.
Pattern: It is a web mining module for NLP and machine learning.
TextBlob: It provides an easy interface to learn basic NLP tasks like sentiment analysis, noun phrase extraction, or pos-tagging.
Quepy: Quepy is used to transform natural language questions into queries in a database query language.
SpaCy: SpaCy is an open-source NLP library which is used for Data Extraction, Data Analysis, Sentiment Analysis, and Text Summarization.
Gensim: Gensim works with large datasets and processes data streams.

02 Tokenization of Text

breaking down a text paragraph into smaller chunks such as words or sentences

Sentence Tokenization	• breaks text paragraph into sentences
Word Tokenization	• breaks text paragraph into words.
Stopwords Removal	• Stopwords considered as noise in the text. • Text may contain stop words such as is, am, are, this, a, an, the, etc.
Part of speech tagging	• Used to filter out a certain figure of Speech • Default is nouns
Stemming and Lemmatization	• converting a word to its base form

03 Word Embeddings : TF-IDF Vectorization

Term Frequency => the frequency of a word in a document

$$TF(\mathbf{term}) = \frac{Number\ of\ times\ \mathbf{term}\ appears\ in\ a\ document}{Total\ number\ of\ items\ in\ the\ document}$$

Inverse Document Frequency => measures the importance of the word in the corpus

- measures how common a particular word is across all the documents in the corpus.

$$IDF(\mathbf{term}) = \log\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ \mathbf{term}\ in\ it}\right)$$

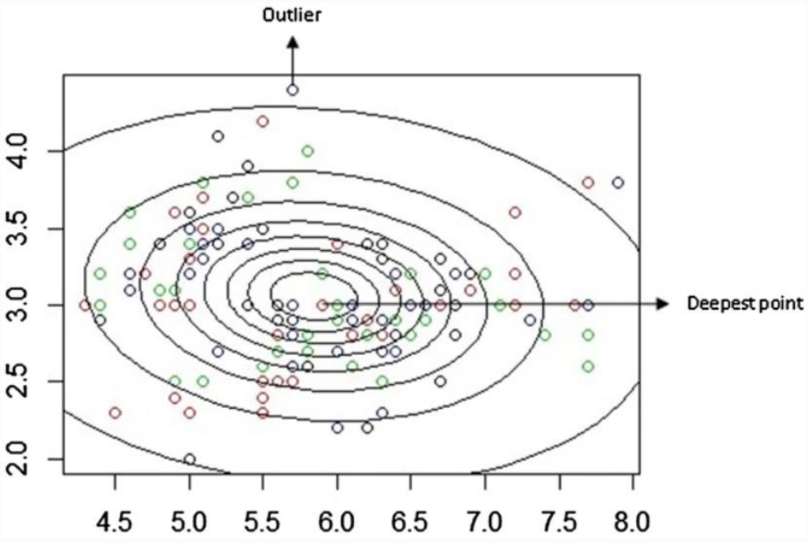
- **Common words would have lesser importance**

$$TFIDF(\mathbf{term}) = TF(\mathbf{term}) * \mathrm{IDF}(\mathbf{term})$$

- TF-IDF manages to incorporate the **significance of a word**. The higher the score, the more important that word is

04 Depth Difference

- Estimates Optimal K value without executing model



Mahalanobis depth contours

The Mahalanobis depth function can be defined as follows:

$$M_D(x; X) = [1 + (x - \bar{x})^T Cov(X)^{-1}(x - \bar{x})]^{-1} \tag{5}$$

where $\bar{x}$ and $Cov(X)$ are the mean and covariance matrix of X, respectively. Maximum depth point is a center point, higher depth value points are near the center, and the lower depth value points are outliers. However, data depth presents globally maximizing depth. Since the mean is sensitive to outliers, the equation to calculate the depth of each point is modified as follows:

$$M_D(x; X_i) = [1 + (x - X_i)^T Cov(X)^{-1}(x - X_i)]^{-1} \tag{6}$$

04 Depth Difference

Algorithm 1 Estimating Number of Clusters

Input: A dataset X with points $X = \{x_1, x_2, \ldots., x_n\}$
Output: k , The number of clusters estimated
$D_i \leftarrow$ Depth of each point in X
$DM \leftarrow$ Depth median of X
$\triangle \leftarrow$ Average difference between D_i and DM
for $k = 2\, to\, 20$ **do**
    $range \leftarrow n/k$
    $start \leftarrow 0$
    $end \leftarrow 0$
    for $j = 1\, to\, k$ **do**
        $start \leftarrow end + 1$
        $end \leftarrow start + range - 1$
        $D_i^k \leftarrow$ Depth of each point within the cluster C_k (partition start : end)
        $DM^k \leftarrow$ Depth median of cluster C_k
        $\triangle^k \leftarrow$ Average difference between D_i^k and DM^k of k^{th} cluster
    end for
    $DW \leftarrow$ Average of $\triangle^k$ of k clusters
    $DB \leftarrow \triangle - DW$
    $DeD \leftarrow DW - DB$
end for
$k \leftarrow index(max(DeD))$ index of DeD for which DeD is maximum

05 Purity Score

Purity is a measure of the extent to which clusters contain a single class.
External Evaluation method which measure show close the clustering is to the predetermined benchmark classes

For each cluster, count the number of data points from the most common class in said cluster. Now take the sum over all clusters and divide by the total number of data points.

Given some clusters M and some set of classes D, both partitioning N data points, purity can be defined as

$$\frac{1}{N}\sum_{m\in M}\max_{d\in D}|m\cap d|$$

This measure doesn't penalize having many clusters, and more clusters will make it easier to produce a high purity. A purity score of 1 is always possible by putting each data point in its own cluster. Also, purity doesn't work well for imbalanced data, where even poorly performing clustering algorithms will give a high purity value.