*Meghnad*

Topic Model

# Technical Document

Topic Model Document Version

|  |  |  |
| --- | --- | --- |
| *Document Version* | *Author* | *Date* |
| *1.0* | *Chiranjeevraja T* | *09.08.2022* |

**Contents**

[1.Introduction 3](#_Toc111019427)

[2.Prerequisites 3](#_Toc111019428)

[3.Config 4](#_Toc111019429)

[*Anchor words* 4](#_Toc111019430)

[4.Data Connectors 5](#_Toc111019431)

[5.Basic Structure of Files 5](#_Toc111019432)

[6.Code Module 6](#_Toc111019433)

[*TopicModelConfig* 6](#_Toc111019434)

[*TopicModelTrn* 6](#_Toc111019435)

[*TopicModelPred* 6](#_Toc111019436)

[7.Output Examples 7](#_Toc111019437)

[*Keywords list* 7](#_Toc111019438)

[*Document list* 7](#_Toc111019439)

[8.Conclusion 7](#_Toc111019440)

# Introduction

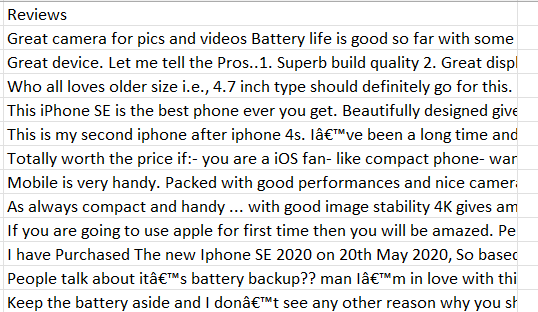
In statistics and natural language processing, a topic model is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents. Topic modelling is a frequently used text-mining tool for the discovery of hidden semantic structures in a text body. Intuitively, given that a document is about a particular topic, one would expect particular words to appear in the document more or less frequently. A document typically concerns multiple topics in different proportions. The "topics" produced by topic modelling techniques are clusters of similar words. A topic model captures this intuition in a mathematical framework, which allows examining a set of documents and discovering, based on the statistics of the words in each, what the topics might be and what each document's balance of topics is.

Correlation Explanation (CorEx) is a topic model that yields rich topics that are maximally informative about a set of documents.CorEx allows a user to integrate their domain knowledge via "anchor words." This integration is flexible and allows the user to guide the topic model in the direction of those words. This allows for creative strategies that promote topic representation, separability, and aspects.

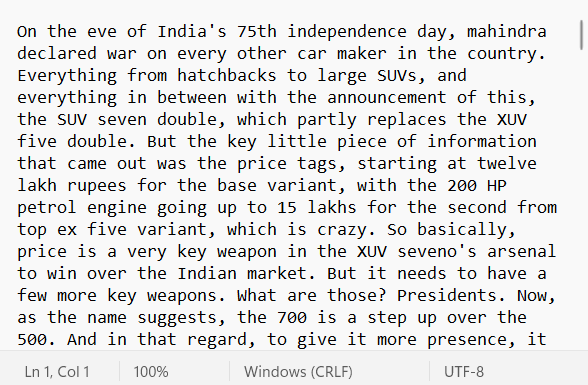
# Prerequisites

The data which is to be trained in this module should follow some necessary formats. This module at present can only read csv and txt.

Sample datasets are as shown below:



**csv file**

****

**txt file**

# Config

Correlation Explanation (CorEx) model is used for topic modeling. The configuration in this module are:

* **stop\_words :** Language of the data
* **max\_features :** Maximum number of words to be trained
* **n\_hidden :** Number of topics
* **anchor\_strength :** Setting anchor strength greater than 5 is strongly enforcing that the CorEx topic model finds a topic associated with the anchor words

# *Anchor words*

Anchored CorEx allows a user integrate their domain knowledge through "anchor words." Anchoring encourages (but does not force) CorEx to search for topics that are related to the anchor words. This helps us find topics of interest, enforce separability of topics, and find aspects around topics. There are a number of strategies we can use with anchored CorEx. Below we provide just a handful of examples.

* ***Anchoring a single set of words to a single topic***

anchors=[['snow', 'cold', 'avalanche']]

* ***Anchoring single sets of words to multiple topics***

anchors=['protest', 'protest', 'protest', 'riot', 'riot', 'riot']

* ***Anchoring different sets of words to multiple topics***

anchors=[['bernese', 'mountain', 'dog'], ['mountain', 'rocky', 'colorado']]

# Data Connectors

* **data\_path:** location of the training data
* **data\_type:** type of the training data ('csv' / 'txt')
* **max\_keyword\_len:** to specify the range of n-gram
* **target\_cols(optional):** attribute names in the data to be used as targets during training
* **anchor\_words(optional):** anchoring provides a way to guide the topic model toward specific subsets of words that the user would like to explore
* **dir\_to\_save\_model(optional**): Directory where the model needs to be saved

# Basic Structure of Files

The structural format of the files and folders present in the text classifier module are as below:

# Code Module

## *TopicModelConfig*

This class belongs to ‘\topic\_model\cfg\config.py’. The details of the functions under this class are as follows,

* ***get\_config(self)***

This function returns default parameters as a dictionary.

## *TopicModelTrn*

This class belongs to ‘\topic\_model\src\trn.py’. The details of the functions under this class are as follows,

* ***config\_connectors(self, data\_path : str, data\_type : str, max\_keyword\_len : int, target\_cols : str = None, anchor\_words : [str] = [],***

***dir\_to\_save\_model : str = None)***

This function sets all the data connector parameters passed from the user.

* ***trn\_ideas(self)***

This function preprocesses and trains the data by the specified parameters, anchor words and returns the model.

* ***\_get\_data\_into\_list(data\_path, data\_type, target\_cols)***

This function processes the data into a list.

* ***\_get\_topic\_list(topics)***

This function processes the topics and returns a dictionary.

## *TopicModelPred*

This class belongs to ‘\topic\_model\src\pred.py’. The details of the functions under this class are as follows,

* ***get\_relevant\_docs(self)***

This function returns a list of documents that are similar to the keyword.

* ***get\_relevant\_keywords(self)***

This function returns list of keywords that are similar to the keyword.

* ***\_get\_topic\_prob\_list(topics)***

This function normalizes the confidence of each keyword and returns as a list.

* ***\_get\_topic\_prob\_dict(topics)***

This function normalizes the confidence of keywords in each topic and returns as a dictionary. It also categorizes the keywords into correlation and inverse-correlation.

* ***\_get\_documents\_list(doc\_id, docs\_list, topics, keyword)***

This function checks which documents contain the keyword and it organizes in an order that first comes the exact matches following the relevant matches

* \_***get\_documents\_dict(lst\_Exact, lst\_Relevant, topic\_lst, docs\_list)***

This function loads all exact and relevant documents and returns them as one dictionary.

# Output Examples

anchor\_words = [["camera","lens","quality"]]

keyword = "camera"

## *Keywords list*

{'Corr': [('camera', 71.14), ('quality', 25.93), ('dual', 0.77), ('sound', 0.53), ('rear', 0.49), ('lens', 0.28), ('average', 0.27), ('picture', 0.23), ('selfie', 0.2), ('mp', 0.17)], 'Inv\_Corr': []}

Corr -> Correlation

Inv -> Inverse Correlation

All anchor words can be seen in the keyword list

## *Document list*

{7337: ('not as good as i thought but battery life is awesome, performance is good and camera quality is f9', [('camera', 58.22), ('quality', 21.23), ('good', 9.83), ('battery', 9.05), ('life', 0.72), ('performance', 0.48), ('ho', 0.33), ('awesome', 0.14)]), 4434: ('Ultimate mobile ... Good Camera ... Good Battery Backup.', [('mobile', 100.0)]),…………,top\_n}

Douments related to the keyword is given as output

# Conclusion

Thus, the topic model is capable of scanning a set of documents, detecting word and phrase patterns within them, and automatically cluster word groups and similar expressions that best characterize a set of documents. This Topic Modeling pipeline enables the user to spend the least effort and get the most effective output.