# Topic Modelling

This code perform Information extraction from the research reports of German public research institutes and perform research analysis such as finding research trends by combining different topic modeling approaches. German research reports were manually annotated to evaluate the different topic modeling techniques.

## Quick Start

Topic Modelling folder contain three files:

- **dataPreprocessing.py**
- **filePreprocessing.py**
- **topicModellingLDA.py**
- **validateResults.py**
- **RunTopicModellingProcess.py**
- **Utility.py**
- **Config.yaml**
- **wordlist-german**

To start using Topic Modelling Code:

### Install Dependencies

Download the following packages using python package installer [pip]

- enchant
- detect
- collections
- translate
- nltk
- gensim
- Pytorch
- Transformers
- Sentence transformers
- Sklearn
- PySpellChecker
- Pandas
- Numpy
- Spacy

### Topic Modelling Code Execution

python RunTopicModellingProcess.py argument

There are multiple modes indicating different stages of the execution. Following are the expected modes and its argument value:

1. Preprocessing
   **Argument:** preprocess

2. LDA
   **Argument:** lda

3. Validation
   **Argument:** validate

**Example:**

python RunTopicModellingProcess.py preprocess

## Overview of files in the Topic Modelling Folder

- **dataPreprocessing.py**

> *Preprocessing Document class perform several pre processing steps to clean the textual data and make it ready to feed data to the model. All the functions of dataPreprocessing class along with their functionality are described below:*

- **init()**: This function set up directory path which contain full extracted text of documents.
- **listAllDocs()** : This function returns list of all documents in the base path
- **tokenArray()**: This function reads the content of the document,preprocess it and detect the language of document
- **removeUnnecessaryWords()**: This function remove the Unnecessary words like Proper Nouns, Adjective etc. from the text
- **preProcessing()**: This function reads the content of the document, preprocess it and detect the language of document
- **remove_numbers()**: This function removes the digits from the content ofthe document
- **spaces()**: This function removes the new line character from the contentof the document
- **remove_punctuation()**: This function removes the punctuations from the content of the document. Underscore "-" was removed from the list of punctuations else the topics of documents contain miss spelled words
- **remove_whitespace()**:This function removes the whitespace from the content of the document
- **specialCharector()**: This function removes the special character from the content of the document
- **removeThreechar()**: This function removes the words less than three character long from the content of the document
- **tokens()**: This function tokenize the content of the document
- **removeStopwordsEnglish()**: This function removes the English stopwords from the preprocessed tokenize text of the document. We also extendedthe NLTK stopword list with new stop words which occurred commonly inthe topic modelling results.
- **removeStopwordsGerman()**: This function that removes the German stopwords from the preprocessed tokenize text of the document. We also extended the NLTK stopword list with new stop words which occurred commonly inthe topic modelling results.
- **removeDuplicateTokens()**: This function that removes the duplicate tokens from the preprocessed tokenize text of the document.

- **fileProcessing.py**

> *The FileProcessing class is used to perform all the Preprocessing Steps on each document and saves the results in a new file for further processing in LDA step. All the functions of fileProcessing class along with their Functionality is described:*

- **init()**:Setup parameters like original documents directory and output directory path. Further it initializes the Spacy module with the pretrained model.
- **removeUnnecessaryWords()**:Function to remove the Unnecessary words like Proper Nouns, Adjective etc. from the text
- **cleanData()**:This function cleans the data using different techniques and methods defined in the dataPreprocessing class and returns the cleaned data for processing

- **xmlStreamParsing():**This function is used to parse the data from the xml file using streams and returns the cleaned data
- **documentsParsing():**This function is used to parse the data from the files and returns the cleaned data
- **mainProcess():**The function first extracts all the documents from the directory, apply all the preprocessing steps and finally save the preprocessed text into a new file with the same file name and in a new directory.

❖ **topicModellingLDA.py**

> *TopicModelling class transform preprocessed textual tokens of document into embeddings, perform k-means clustering and finally use LDA to find topics of the document. All the functions of topicModellingLDA class along with their Functionality is described:*

- **init():** This function set up parameters like tokenizer type, embeddings model, filename to save topics, number of clusters, number of topics and number of words in each topic.
- **set_model_type_multilingual_bert():** This function set up specific configurations in order to transform preprocessed textual tokens into word embeddings using Multilingual Bert model
- **set_model_type_dmbz():** This function set up specific configurations in order to transform preprocessed textual tokens into word embeddings using MDZ Digital Library German BERT model
- **set_model_type_electra():** This function set up specific configurations in order to transform preprocessed textual tokens into word embeddings using German Electra BERT model
- **set_model_type_gottbert():** This function set up specific configurations in order to transform preprocessed textual tokens into word embeddings using Gott BERT model
- **transform_tokens_to_embeddings():** This function read the preprocessed textual tokens, convert the tokens into batch of 512 tokens, transform the tokens into word embeddings and finally concatenated all the word embeddings together
- **batch():** This generator function that divide the iterable into chunksand yield chunk of specific length of iterable
- **assigned_clusters():** This function cluster the word embeddings using k-means clustering
- **seperate_clusterwords():** This function separate words assigned to each cluster
- **fit_lda_model():** This function uses the LDA model to find topic in each cluster of words
- **clean_results():**This function cleans the resultant topic in two ways, first by Removing any duplicate values and then Removing the words that are not part of German Dictionary
- **mainProcess():** This is the main function that list all the documents, preprocessed documents text, transform text of document into word embeddings, perform k-means clustering and finally use LDA to findtopics of the document

❖ **ValidateResults.py**

> *The validateResults class validates the results of LDA step by checking if the words are valid/dictionary word and does these words make sense. All the functions of validateResults class along with their Functionality is described:*

- **init():** Setup parameters like filename, input_file path and output file path.Further it also initializes the Pyspellchecker for checking correctness of each words
- **clean_results():**This function cleans the resultant LDA topic in three ways, first it

Removes any duplicate values, then it checks if any compound word has occurred and if so, it checks it is in proper form or not and then finally, remove the words that are not part of German Dictionary

- **extractCompletedFiles():** This function will first extract all the files that are already processed and return that in a list
- **mainProcess():** This function first reads the LDA's results from a file and then validate each result using the above mentioned methods.

❖ **Config.yaml**

> *This YAML file contains all the configuration parameters to run the whole topic modelling functionality. All the parameters and there purpose is described:*

- **preprocessing.input_dir:** The documents directory which has to be preprocessed.
- **preprocessing.output_dir:** The output directory where preprocessed files has to be saved.
- **preprocessing.xml_dump_path:** The location of the XML file which has to be preprocessed.
- **preprocessing.process_xml_dump:** Boolean value indicating whether to preprocess XML file of any other type of file
- **lda.input_dir:** The preprocessed documents directory on which LDA has to be applied.
- **lda.output_dir:** The output directory where LDA topics for each document is saved.
- **lda.mode:** This indicates which approach to apply for extracting the topics from a document. There are currently 3 approaches: Word Embedding's Only, LDA Only and Word Embedding and LDA Combined.
- **validation.input_dir:** The LDA result file location on which validation is to be applied.
- **validation.output_dir:** The output directory where validated LDA topics for each document is saved.
- **validation.lda_filename:** The file name of the LDA result file.

❖ **RunTopicModellingProcess.py**

> *This file simply import the TopicModelling class from topicModellingLDA.py file, set up the word embedding model to be used for topic modelling and Call the mainProcess function of TopicModelling class to find topics of the document.*

### Order of function execution to find topics of the documents is as follows:

First we call the mainProcess() function of TopicModelling class. This function then list all the documents using **listAllDocs()** function of the Preprocessing Documents Class. It then calls the functions in following order to find topics of each document:

> *1)* ***tokenArray():*** *Perform several pre processing steps to clean the textual data and make it ready to feed data to the model*

> *2)* ***transform_tokens_to_embeddings():*** *Transform textual data of document into word embeddings*

> *3)* ***assigned_clusters():*** *Perform k-means clustering on word embeddings*

> *4)* ***seperate_clusterwords():*** *Separate list of words that exist in specific cluster*

> *5)* ***fit_lda_model():*** *Fit the lda model on list of words that exist inside the specific cluster and get topic of that specific cluster*

> *6)* ***clean_results():*** *Cleans the results from the LDA by removing the duplicate words and removing words that are not part of German Dictionary.*

> *7)* ***evaluation_document_topics():*** *Evaluate and save the topic modelling results*