Low Level Design

News Articles Sorting

Written By	Ujjwal Kar
Document Version	0.23
Last Revised Date	30.08.2022

Contents

1. Introduction

1.1. What is a Low-Level design document?

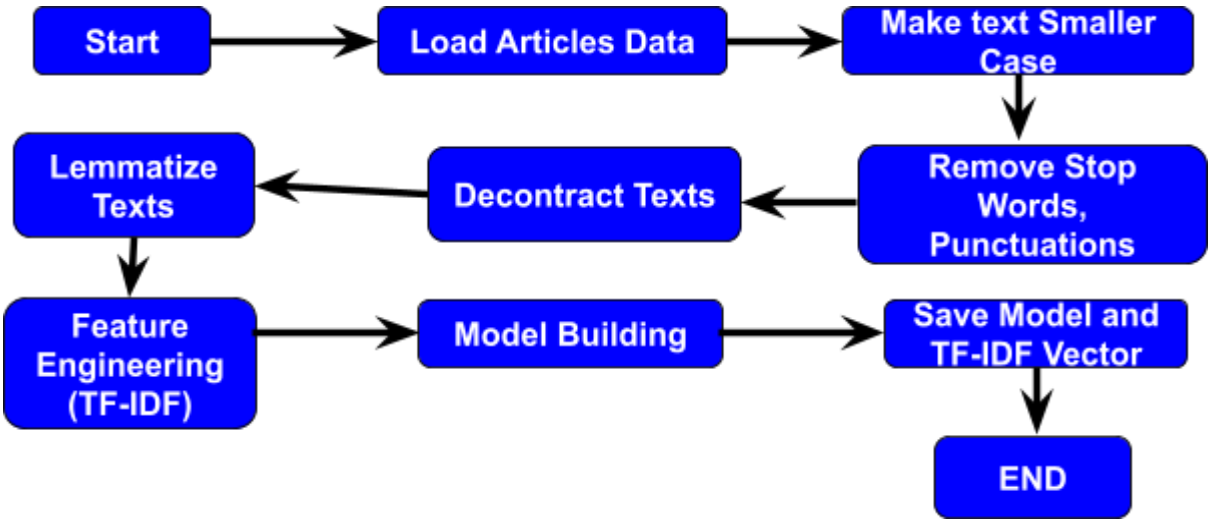
The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for the News Articles Sorting. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

1.2. Scope

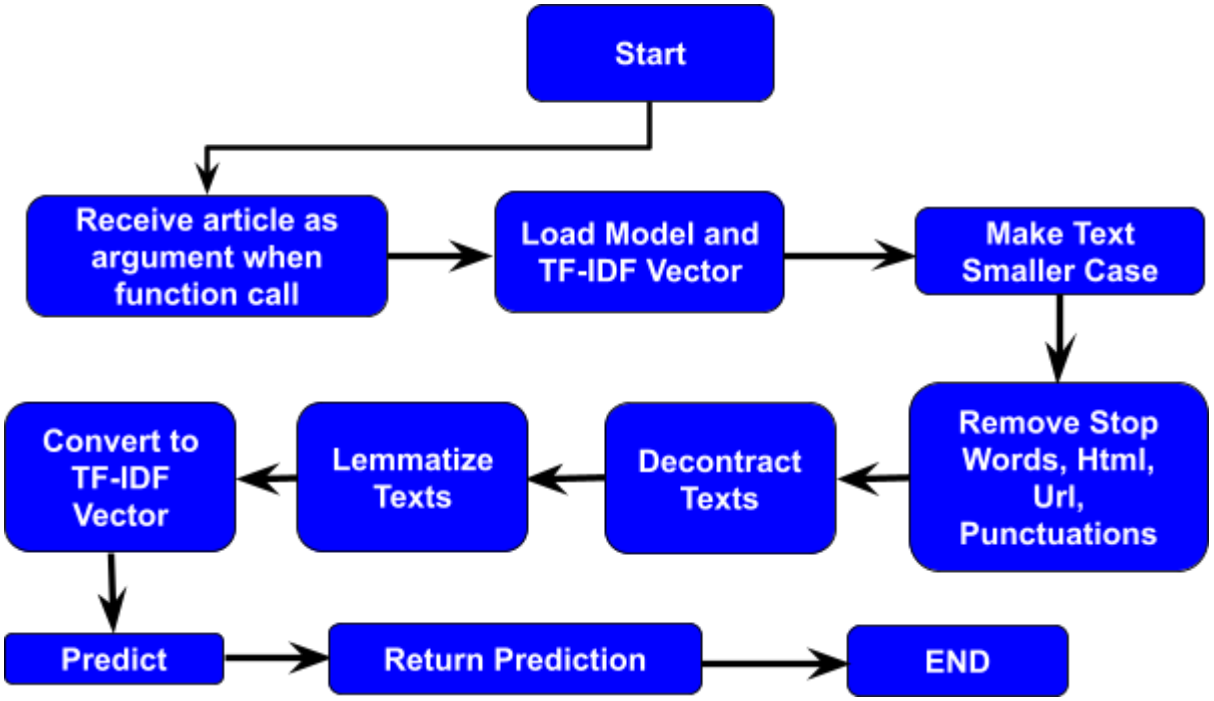
Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code, and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work

2. Architecture

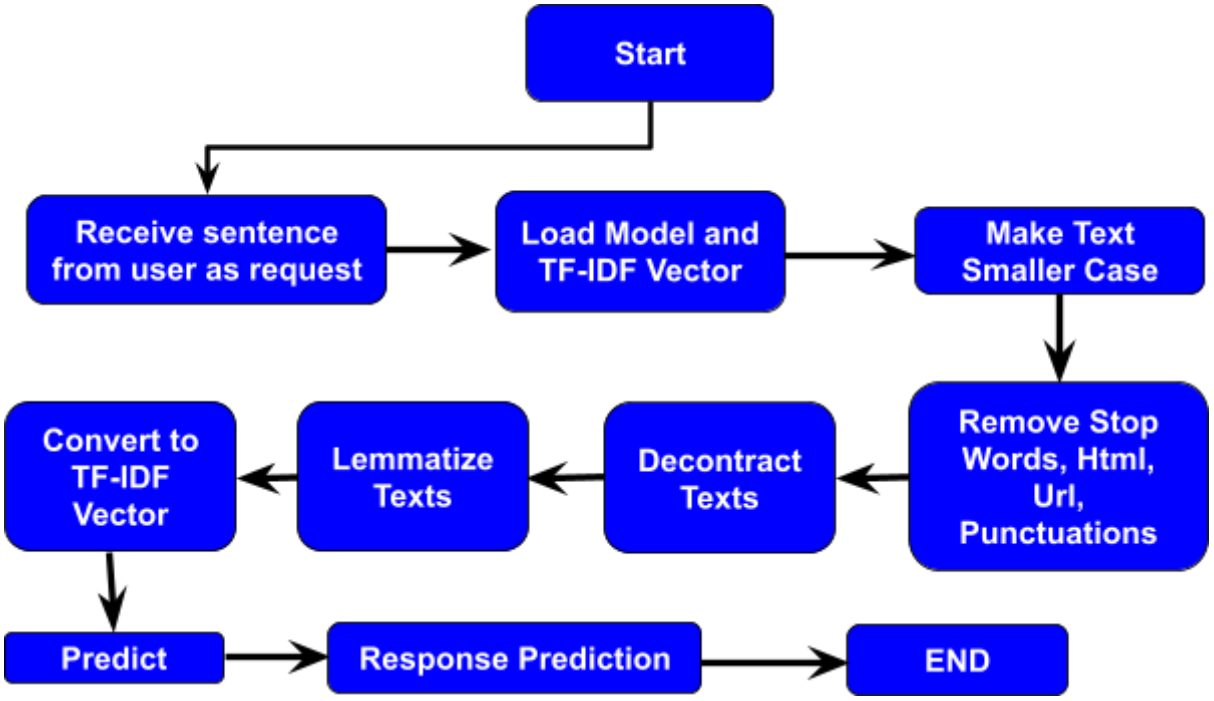
2.1. Training Model and Save



2.2. Python Module



2.3. Rest API


Start
Receive sentence from user as request
Load Model and TF-IDF Vector
Make Text Smaller Case
Remove Stop Words, Html, Url, Punctuations
Decontract Texts
Lemmatize Texts
Convert to TF-IDF Vector
Predict
Response Prediction
END

3. Architecture Description

3.1. Load Articles Data

Load BBC News Classification dataset from Kaggle, containing 1490 articles with their Category, in CSV format.

3.2. Make Text Smaller case

Make all words to the smaller case as "A" and "a" have different meanings for computers, but they are the same.

3.3. Remove Stop Words and Punctuations

Words that frequently occur in sentences and carry no significant meaning in sentences. These are not important for prediction, so we remove stopwords to reduce data size and prevent overfitting.

3.4. Decontracting Texts

Contraction is shorting words made by putting two words together. In decontraction, we split this short word into two words.

3.5. Lemmatizing Texts

Lemmatization is converting words or synonyms into their root word using vocabulary mapping.

3.6. Feature Engineering(TF-IDF)

Term frequency (TF): Number of times a term has appeared in a document. The term frequency is a measure of how frequently or how common a word is for a given sentence.

Inverse Document Frequency (IDF): The inverse document frequency (IDF) is a measure of how rare a word is in a document. Words like “the”,” a” show up in all the documents but rare words will not occur in all the documents of the corpus.

If a word appears in almost every document means it’s not significant for the classification.

IDF of a word is = log(N/n)

N: total number of documents.
n: number of documents containing a term (word)

TF-IDF Evaluates how relevant is a word to its sentence in a collection of sentences or documents.

3.7. Model Building and save model and TF-IDF vector

Among any other algorithm complement naive Bayes gives better results in terms of accuracy, precision, recall, f1-score, and support. So we use complement naive Bayes for prediction. Then we save our model and TF-IDF vector as a pickle.

3.8. Receive a sentence from the user as a request or argument and load models and TFIDF Vector

A REST API is built for users to make a post request with their article. From the backend of the website, we receive the sentence.
A python module is also built for the user to call a function with the article as an argument.
After that loading model and TFIDF Vector, we can predict the category of articles.

3.9. Text Processing on Received Article

Makes Text Smaller Case Remove Stop Words, Html, Url, Punctuations Decontract Texts Lemmatize Texts Convert to TF-IDF Vector

3.10. Predict the Category of the article and send it as a response (REST API) or return prediction(Python Module).

4. Unit Test Cases

Test Case Description	Pre-Requisite	Expected Result
Verify whether the application URL is accessible to user	1. Application URL should be defined	Application URL should be accessed by the user
Verify whether the application loads completely for the user when the URL is accessed	1. Application URL is accessible 2. Application is deployed	The Application should load completely for the user when the URL is accessed
Verify whether the POST request is done on REST API.	1. Application URL is accessible 2. User makes a post request with an article on URL	The Application should Response the category of Article.
Verify whether the python module is loadable	1. The module name should be defined 2. Python and pip are installed on the computer.	Python module should be accessed by the user
Verify whether the python module is installable	1. The module name should be defined 2. Python and pip are installed on the computer.	Python module should be installed by the user
Verify whether the classification model and TF-IDF vector were downloaded with the python module	1. Install python module 2. Article classification function call.	Load classification model and TF-IDF vector.
Verify whether the module can predict.	1. Install the python module 2. Article classification function call. 3. Pass an article as an argument on function call.	The module should return the category of Article.