Interpretation of the model and parameters:

tfidf\_model = TfidfVectorizer(stop\_words=stop\_words, ngram\_range=(2, 10),

analyzer='word', binary=False,

token\_pattern=r"\w+", sublinear\_tf=False)

stop\_words: string {'english'}, list, or None(default). If English is used, a list of stop words is integrated into the language; If list is specified, all words in the list are assumed to be stop words, and they will be eliminated from the token; There are no deactivation words used if None is selected.

ngram\_range: tuple(min\_n, max\_n). The lower and upper range of n-values of the n-gram to be extracted, for all values of n in the min\_n <= n <= max\_n interval.

analyzer: string, {'word', 'char'} or callable. Defines a feature as a word or n-gram character if the call passed to it is used to extract a sequence of features from an unprocessed input source file;

binary: boolean, False by default. If not True, all non-zero counts are set to 1, which is useful for discrete probability models that model binary events instead of integer counts;

token\_pattern: string. The regular expression shows the composition of the "token" and is only used if analyzer == 'word'. Regular expression for two or more alphanumeric characters (punctuation is completely ignored and always treated as a token separator).

sublinear\_tf: boolean, optional. Apply a linear scaling TF, e.g., using 1+log(tf) to override tf.

The word frequency matrix is then transformed into a TF-IDF weight matrix using the fit transform function, and the weight matrix is sorted to get the keywords corresponding to the top 10 TF-IDF values.