Tags Generator for Legal Text Documents

*Data Science Project*

2021

Rahul Sharma

Prasfur Tiwari

2/19/2021

**Index**

**Introduction 3**

**Problem Statement 3**

**Data Analysis 3**

**Pre-processing 4**

**Data Cleaning 4**

**Lemmatization 5**

**Vectorization 6**

**Data Visualization 6**

**Data Modeling 7**

**Results 8**

**Prediction 8**

**Tags Generator for Legal Text Documents**



**Introduction:**

Tags have been a relevant part of information retrieval system. They help the system to get relevant information in most effective way. These tags have been generated manually by human annotators and makes the system more expensive for the clients. The only way to make these technologies affordable is to find a way to automatically generate tags that resembles to human annotations

**Problem Statement:**

We are provided with a legal text document in Training set documents. In Training Tag documents, the relevant tags have been manually created by human annotators respectively to the files of Training set documents. The aim of this project is to create a solution that automatically generate tags for the Testing set legal text document.

**Data Analysis:**

The data used was provided by a law firm. The data on which the model is trained consists of the tags used by a functioning law firms for the particular text.

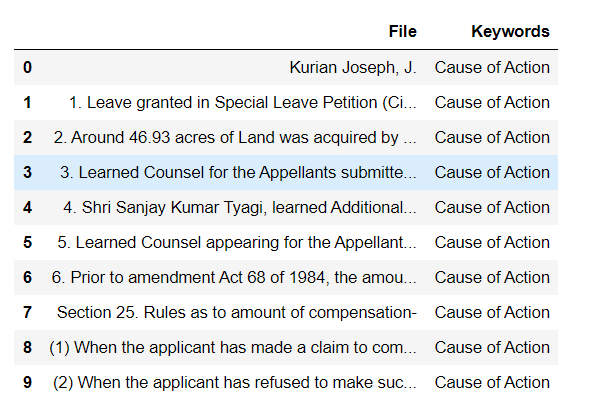
**Data Pre-processing:**

Since the data was present in individual text files, the first task was to store the data from an individual text file into a single data frame. On performing the required code, the following was obtained:

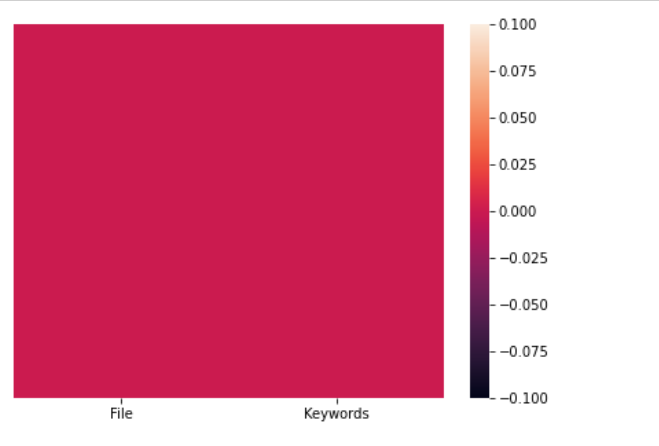


**Data Cleaning:**

As it is visible from the above image, the data in the “File” column contains \n, which can affect the accuracy of the model. Hence, “\n” was replaced and the final outcome of the dataset is as follows:



**Heatmap:** To identify null values:

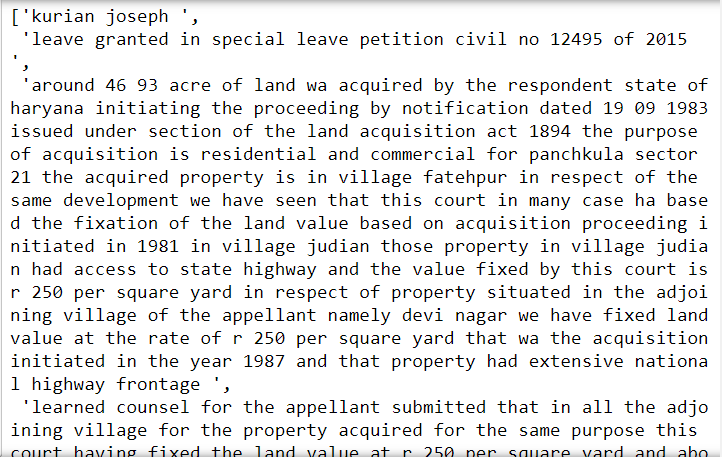


Since the heatmap is uniform, this represents the presence of **no null values** in the dataset.

**Data Lemmatization**:

The “File” column of the data frame contains a lot of stop words and words which are irrelevant for model creation.

The pre-processing of the data is performed by using the libraries re and nltk. The unnecessary elements have been removed to produce an accurate model, using Lemmatization.



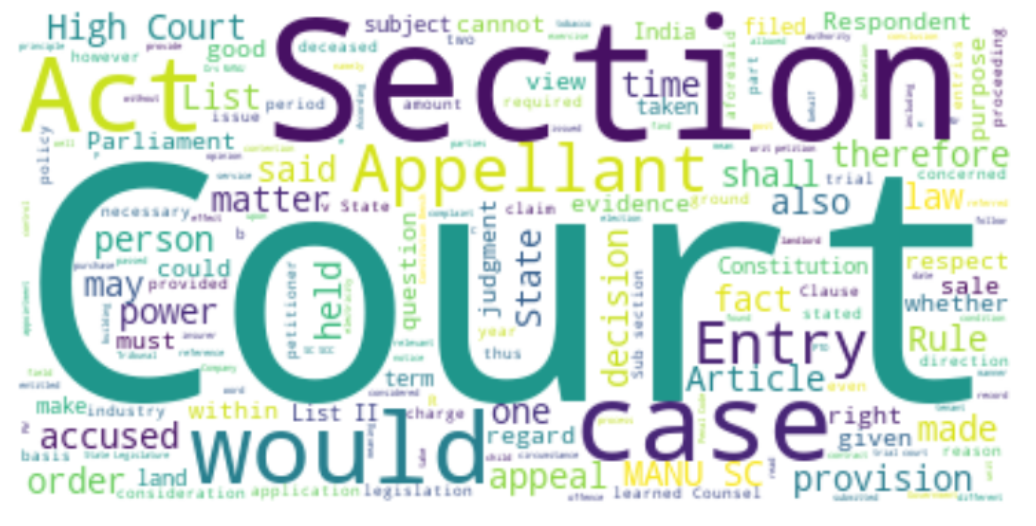
**Data Vectorization:**

Since our model needs numeric data, we need to transform the processed sentences into vectors.

To convert the above processed text into vectors, TF-IDF vectorizer is preferred over Bag Of words (since TF-IDF is more accurate, especially when dealing with a project in which keyword generation is the key objective to be achieved.)

**Data Visualization:**

**Word cloud** for training data:



From the below visualization, we can see that the most common words used in the file column of the dataset are:

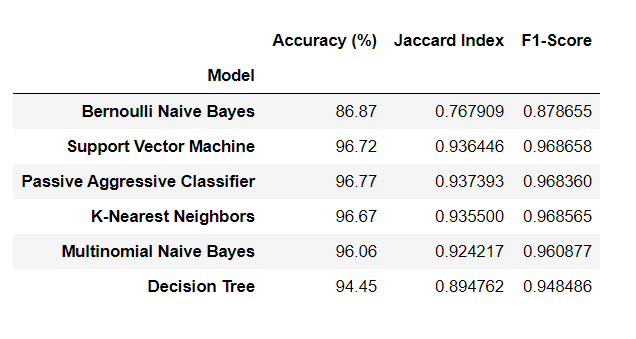
* **Court**
* **Section**
* **Act**
* **Appellant**
* **Case**

**Model Fitting:**

These are the algorithms used in this project:

1. [**Bernoulli’s Naive B**](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html#sklearn.naive_bayes.BernoulliNB)**ayes**: It implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable.
2. **Support Vector Machine:** In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis.
3. **Passive Aggressive Classifier:** The Passive-Aggressive algorithms are a family of Machine learning algorithms that are not very well known by beginners and even intermediate Machine Learning enthusiasts. However, they can be very useful and efficient for certain applications.
4. ***k*-Nearest Neighbour:** It is a type of [classification](https://en.wikipedia.org/wiki/Classification), where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then [normalizing](https://en.wikipedia.org/wiki/Normalization_(statistics)) the training data can improve its accuracy dramatically.
5. **Multinomial Naive Bayes classifier**: It is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts.
6. **Decision Trees: They** are a non-parametric supervised learning method used for [classification](https://scikit-learn.org/stable/modules/tree.html#tree-classification) and [regression](https://scikit-learn.org/stable/modules/tree.html#tree-regression). The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

**Results:**



Since the **SVM** model shows the best results on the training set i.e. **96.72%**, along with comparatively better F-1 Score, we choose it for predicting the keywords for the test set.

**Prediction:**

The same SVM model is used for the prediction of the test set’s keywords.

The data from the text set cleaned, lemmatized, vectorized and then passed in the model. The generated keywords are then exported to a separate excel file.

