1.- IMPLEMENTACIÓN:

1.1.- Programa Principal(main):

Para la realización de este proyecto se ha optado por la implementación de un fichero “main.cpp” muy sencillo donde simplemente se llamará al programa con una opción determinada y dependiendo de esta opción se ejecutará una de las funciones principales del proyecto, ya se al a de crear el vocabulario o la de clasificar entre otras.

```
/**
 * @brief      Función Main del programa recibe el fichero de datos como
 *             parametro.
 *
 * @param[in]  argc  El número de argummentos.
 * @param      argv  El array de argumentos
 *
 * @return     0 Si el programa finaliza correctamente.
 */
int main (int argc, char* argv[]) {
    if (argc <= 1) {
        printHelp();
    }
    else {
        std::string flag = argv[1];
        if (flag == "-h" || flag == "--help") {
            printHelp();
            return(0);
        }
        else if (flag == "-v" || flag == "--vocabulary") {
            generateVocabulary(argc, argv);
        }
        else if (flag == "-co" || flag == "--corpus") {
            generateCorpus(argc, argv);
        }
        else if (flag == "-l" || flag == "--learner") {
            generateLearner(argc, argv);
        }
        else if (flag == "-cl" || flag == "--classify") {
            generateClassifier(argc, argv);
        }
        else if (flag == "-e" || flag == "--error") {
            calculateError(argc, argv);
        }
    }

    std::cout << std::endl << "Program finished correclty." << std::endl;
    return 0;
}
```

Todas estas funciones tienen en común que utilizan las clases Token y preProcessor para llevar a cabo su correcto funcionamiento por lo que serán las clases que comentemos a continuación.

Gabriel Melián Hernández
alu0100819786
alu0100819786@ull.edu.es

1.2.- Clase Token.

La clase Token es aquella que tendrá todos los métodos necesarios para poder almacenar y manipular la información de cada una de las palabras que estarán presentes en nuestro vocabulario. Almacenaremos su nombre, la cantidad de veces que aparece en el corpus, su probabilidad, el tipo de token que es y la probabilidad de que aparezca para cada una de las clases que disponemos (En este caso, Negative o Positive).

Disponemos también de todos los métodos Setters y Getters para controlar estos atributos principales de los Token, además de sobrecarga de todos los operadores para que las operaciones se ejecuten de la forma correcta.

Por último tendremos una función “incrementate” que irá sumando 1 a la cantidad de cada token siempre que aparezcan y dos funciones para la probabilidad, una para la clase y otra para generar la probabilidad con el logaritmo aplicando suavizado laplaciano para aquellas palabras que tengan 0 apariciones en algún corpus.

```
/**
 * @brief      Genera la probabilidad de un token.
 *
 * @param[in]  vocSize       Tamaño del vocabulario.
 * @param[in]  tokenAmmount  Cantidad de veces que aparece un token
 *
 * @return     El Log de la probabilidad.
 */
float Token::generateLogProb (const unsigned& vocSize, const unsigned& tokenAmmount) {
    probability_ = (ammount_ + 1);
    int divider = (vocSize + tokenAmmount);
    probability_ /= divider;
    return std::log(probability_);
}
```

Gabriel Melián Hernández
alu0100819786
alu0100819786@ull.edu.es

```cpp
class Token {

    private:
        // Atributos
        std::string name_;              // El nombre del token.
        unsigned ammount_;              // La cantidad de veces que aparece el token en el corpus.
        float probability_;             // La probabilidad.
        std::string type_;              // El tipo del token en las clases especificadas.
        std::vector<float> multiClass_; // La probabilidad del vector para la clasificación.

    public:
        // Constructores y Destructor.
        Token (void);
        Token (std::string name);
        ~Token (void);

        // Setters y Getters.
        std::string get_Name(void) const;
        unsigned get_Ammount (void) const;
        float get_Probability (void) const;
        float get_MultiClass (unsigned pos) const;
        std::vector<float> get_MultiClass (void) const;
        std::string get_Type (void) const;

        void set_Name (std::string name);
        void set_Ammount (unsigned ammount);
        void set_Probability (float newProbability);
        void set_MultiClass (std::vector<float> newMultiClass);
        void set_Type (std::string newType);

        // Sobrecarga de Operadores
        bool operator< (const Token& otherToken) const;
        bool operator<= (const Token& otherToken) const;
        bool operator< (const std::string& str) const;
        bool operator<= (const std::string& str) const;
        bool operator> (const Token& otherToken) const;
        bool operator>= (const Token& otherToken) const;
        bool operator> (const std::string& str) const;
        bool operator>= (const std::string& str) const;
        bool operator== (const Token& otherToken) const;
        bool operator== (const std::string& str) const;
        Token& operator++ (void);
        Token& operator= (const Token& otherToken);

        // Funciones.
        void incrementate (void);
        void addClassProb (float prob, std::string newClass);
        float generateLogProb (const unsigned& vocSize, const unsigned& tokenAmmount);

        // Escritura.
        void printToken (void) const;
```

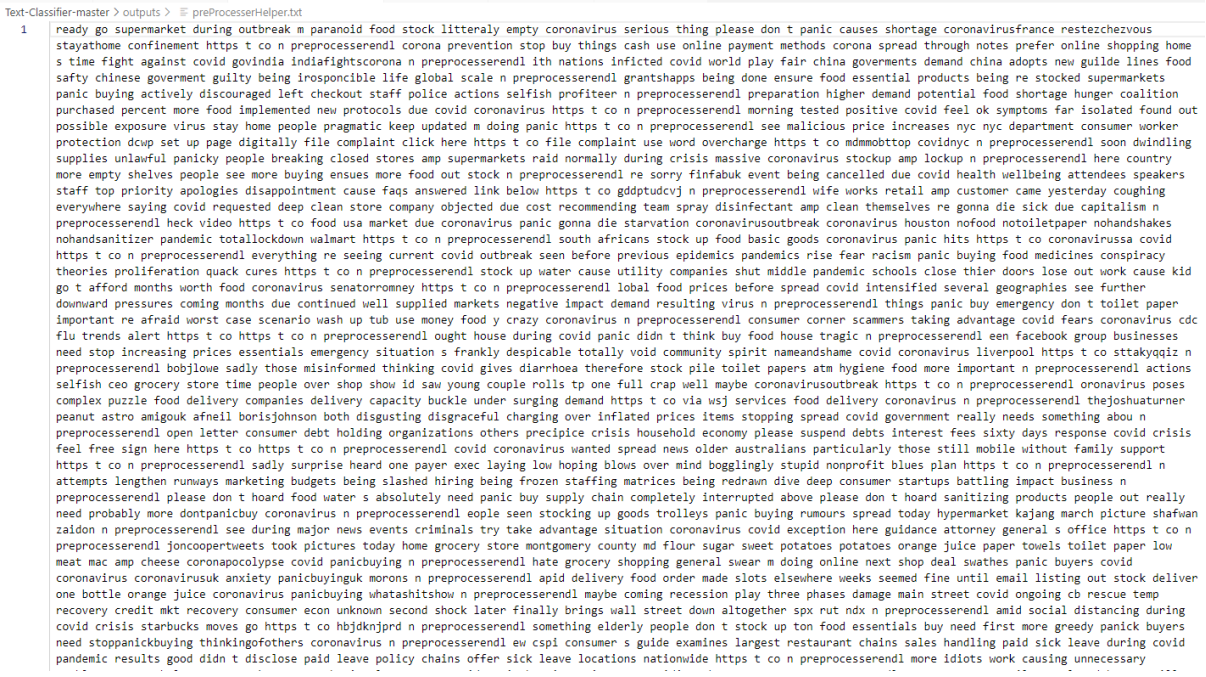
Gabriel Melián Hernández
alu0100819786
alu0100819786@ull.edu.es

1.3.- Clase PreProcesser:

Para generar un buen vocabulario, antes tenemos que realizar diversas tareas de pre procesado de los datos de entrada, de esta manera conseguiremos reducir notablemente el tamaño de nuestro vocabulario y dejar solamente aquellas palabras que tengan significado. Algunas de estas tareas de PreProcesado son las de convertir todo a minúsculas o mayúsculas, eliminar las stopWords, eliminar los diferentes signos de puntuación, las URLS, los hashtags o los números.

Al tratarse de tweets nuestros datos de entrada, tenemos que tener en cuenta que mucha de la información que vamos a recibir es irrelevante para nuestro Clasificador como pueden ser las cuentas de usuarios (@XXXXX) los enlaces a diferentes webs etc… Por lo tanto se vuelve necesaria la realización de este tipo de tareas antes de realizar ningún otro paso.

Una vez realizadas estas tareas de PreProcesser, se almacena para su futuro uso los resultados en un fichero de texto que contendrá todo lo necesario para llevar a cabo la correspondiente creación del vocabulario etc…


Text-Classifier-master > outputs > preProcesserHelper.txt

1 ready go supermarket during outbreak m paranoid food stock litteraly empty coronavirus serious thing please don t panic causes shortage coronavirusfrance restezchezvous stayathome confinement https t co n preprocesserendl corona prevention stop buy things cash use online payment methods corona spread through notes prefer online shopping home s time fight against covid govindia indiafightscorona n preprocesserendl ith nations inficted covid world play fair china goverments demand china adopts new guilde lines food safty chinese goverment guilty being irosponcible life global scale n preprocesserendl grantshapps being done ensure food essential products being re stocked supermarkets panic buying actively discouraged left checkout staff police actions selfish profiteer n preprocesserendl preparation higher demand potential food shortage hunger coalition purchased percent more food implemented new protocols due covid coronavirus https t co n preprocesserendl morning tested positive covid feel ok symptoms far isolated found out possible exposure virus stay home people pragmatic keep updated m doing panic https t co n preprocesserendl see malicious price increases nyc nyc department consumer worker protection dcwp set up page digitally file complaint click here https t co file complaint use word overcharge https t co mdmmobttop covidnyc n preprocesserendl soon dwindling supplies unlawful panicky people breaking closed stores amp supermarkets raid normally during crisis massive coronavirus stockup amp lockup n preprocesserendl here country more empty shelves people see more buying ensues more food out stock n preprocesserendl re sorry finfabuk event being cancelled due covid health wellbeing attendees speakers staff top priority apologies disappointment cause faqs answered link below https t co gddptudcvj n preprocesserendl wife works retail amp customer came yesterday coughing everywhere saying covid requested deep clean store company objected due cost recommending team spray disinfectant amp clean themselves re gonna die sick due capitalism n preprocesserendl heck video https t co food usa market due coronavirus panic gonna die starvation coronavirusoutbreak coronavirus houston nofood notoiletpaper nohandshakes nohandsanitizer pandemic totallockdown walmart https t co n preprocesserendl south africans stock up food basic goods coronavirus panic hits https t co coronavirussa covid https t co n preprocesserendl everything re seeing current covid outbreak seen before previous epidemics pandemics rise fear racism panic buying food medicines conspiracy theories proliferation quack cures https t co n preprocesserendl stock up water cause utility companies shut middle pandemic schools close thier doors lose out work cause kid go t afford months worth food coronavirus senatorromney https t co n preprocesserendl lobal food prices before spread covid intensified several geographies see further downward pressures coming months due continued well supplied markets negative impact demand resulting virus n preprocesserendl things panic buy emergency don t toilet paper important re afraid worst case scenario wash up tub use money food y crazy coronavirus n preprocesserendl consumer corner scammers taking advantage covid fears coronavirus cdc flu trends alert https t co https t co n preprocesserendl ought house during covid panic didn t think buy food house tragic n preprocesserendl een facebook group businesses need stop increasing prices essentials emergency situation s frankly despicable totally void community spirit nameandshame covid coronavirus liverpool https t co sttakyqqiz n preprocesserendl bobjlowe sadly those misinformed thinking covid gives diarrhoea therefore stock pile toilet papers atm hygiene food more important n preprocesserendl actions selfish ceo grocery store time people over shop show id saw young couple rolls tp one full crap well maybe coronavirusoutbreak https t co n preprocesserendl oronavirus poses complex puzzle food delivery companies delivery capacity buckle under surging demand https t co via wsj services food delivery coronavirus n preprocesserendl thejoshuaturner peanut astro amigouk afneil borisjohnson both disgusting disgraceful charging over inflated prices items stopping spread covid government really needs something abou n preprocesserendl open letter consumer debt holding organizations others precipice crisis household economy please suspend debts interest fees sixty days response covid crisis feel free sign here https t co https t co n preprocesserendl covid coronavirus wanted spread news older australians particularly those still mobile without family support https t co n preprocesserendl sadly surprise heard one payer exec laying low hoping blows over mind bogglingly stupid nonprofit blues plan https t co n preprocesserendl n attempts lengthen runways marketing budgets being slashed hiring being frozen staffing matrices being redrawn dive deep consumer startups battling impact business n preprocesserendl please don t hoard food water s absolutely need panic buy supply chain completely interrupted above please don t hoard sanitizing products people out really need probably more dontpanicbuy coronavirus n preprocesserendl eople seen stocking up goods trolleys panic buying rumours spread today hypermarket kajang march picture shafwan zaidon n preprocesserendl see during major news events criminals try take advantage situation coronavirus covid exception here guidance attorney general s office https t co n preprocesserendl joncoopertweets took pictures today home grocery store montgomery county md flour sugar sweet potatoes potatoes orange juice paper towels toilet paper low meat mac amp cheese coronapocolypse covid panicbuying n preprocesserendl hate grocery shopping general swear m doing online next shop deal swathes panic buyers covid coronavirus coronavirusuk anxiety panicbuyinguk morons n preprocesserendl apid delivery food order made slots elsewhere weeks seemed fine until email listing out stock deliver one bottle orange juice coronavirus panicbuying whatashitshow n preprocesserendl maybe coming recession play three phases damage main street covid ongoing cb rescue temp recovery credit mkt recovery consumer econ unknown second shock later finally brings wall street down altogether spx rut ndx n preprocesserendl amid social distancing during covid crisis starbucks moves go https t co hbjdknjprd n preprocesserendl something elderly people don t stock up ton food essentials buy need first more greedy panick buyers need stoppanickbuying thinkingofothers coronavirus n preprocesserendl ew cspi consumer s guide examines largest restaurant chains sales handling paid sick leave during covid pandemic results good didn t disclose paid leave policy chains offer sick leave locations nationwide https t co n preprocesserendl more idiots work causing unnecessary

Gabriel Melián Hernández
alu0100819786
alu0100819786@ull.edu.es

```
private:
    // Atributos.
    std::string inputFile_;                    // Fichero de entrada
    std::string outputFile_;                   // Fichero de salida donde se guarda el preProceso.
    std::string data_;                         // datos leidos del fichero de entrada.

public:
    // Constructores & Destructor.
    PreProcesser (void);
    PreProcesser (std::string inputFile, std::string outpuFile);
    ~PreProcesser (void);

    // Getters & Setters
    std::string get_InputFile (void) const;
    std::string get_OutputFile (void) const;
    std::string get_Data (void) const;

    void set_InputFile (std::string newInputFile);
    void set_OutputFile (std::string newOutputFile);
    void set_Data (std::string newData);

    // Sobrecarga de Operadores.
    PreProcesser& operator= (const PreProcesser& newPreProcesser);

    // Funciones.
    void convertLowerCase (std::string& str);
    void convertLowerCase (void);
    void convertUpperCase (std::string& str);
    void convertUpperCase (void);
    void eraseReservedWords (std::vector<std::string>& reservedWords, std::string& fileName);
    std::string eraseReservedWords (std::string& sentence, std::vector<std::string>& reservedWords);
    void erasePunctuationSigns (std::string& str);
    void erasePunctuationSigns (void);
    void eraseURLs (std::string& str);
    void eraseURLs (void);
    void eraseHashtags (std::string& str);
    void eraseHashtags (void);
    void eraseNumbers (std::string& str);
    void eraseNumbers (void);
    void eraseAllNumbers (void);
    void eraseAllNumbers (std::string& str);

    // Lectura & Escritura.
    int loadData (std::string& inputFile, std::string dataType);
    void loadTestData (std::string& inputFile);
    void printData (void);
    void storeData(std::string& outputFile, int dataLines);
```

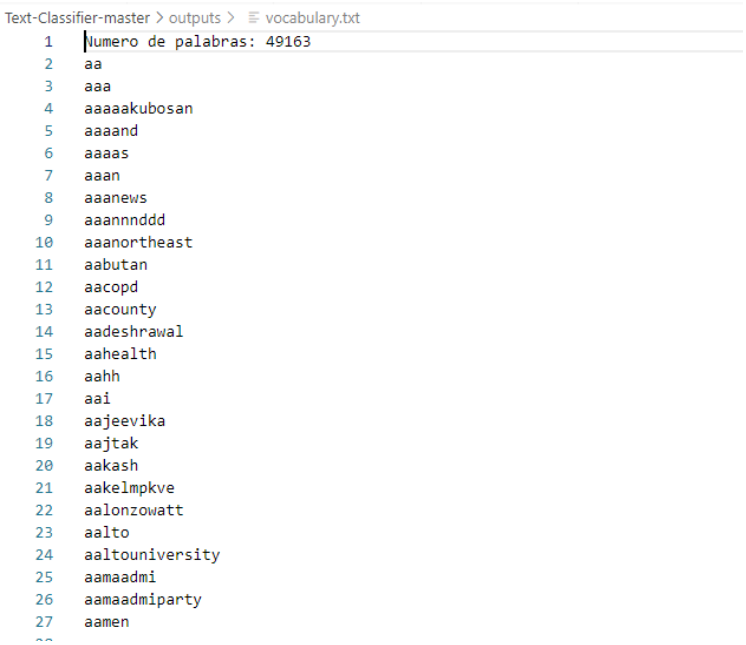
1.4.- Clase Vocabulary.

La clase Vocabulary es la primera que tendrá una función función propia en el programa main y será la encargada de generar y almacenar el vocabulario de nuestros datos de entrada, almacenando el número total de palabras que existen en el vocabulario y cuales son. Para ello usará como dijiste anteriormente las clases PreProcesser y token, además de sus métodos propios.

```
/**
 * @brief      Genera el vocabulario desde el primer fichero de entrada y lo almacena
 *             en el segundo fichero, haciendo uso de las stopWords ddel tercer fichero.
 *
 * @param      argc  El conteo de argumentos.
 * @param      argv  Los argumentos del array.
 */
void generateVocabulary (int& argc, char* argv[]) {
    if (argc != 5) {
        std::cout << std::endl << "Error, the program needs 4 arguments to generate the vocabulary:" << std::endl <<
        exit(1);
    }
    std::string originFile = argv[2];
    std::string outputFile = argv[3];
    std::string reservedWords = argv[4];
    std::string fileHelper = "../outputs/preProcesserHelper.txt";
    Vocabulary vocabulary(originFile, outputFile);
    {
        Chrono chrono;
        chrono.startChrono();
        vocabulary.preProcessData(reservedWords);
        chrono.stopChrono();
        std::cout << std::endl << "Elapsed pre-processing time: " << chrono.get_Seconds(5) << " seconds." << std::enc
    }
    {
        Chrono chrono;
        chrono.startChrono();
        vocabulary.generateVocabulary(fileHelper, false);
        chrono.stopChrono();
        std::cout << std::endl << "Elapsed generating vocabulary time: " << chrono.get_Seconds(5) << " seconds." << s
    }
    {
        Chrono chrono;
        chrono.startChrono();
        vocabulary.storeVocabulary(outputFile);
        chrono.stopChrono();
        std::cout << std::endl << "Elapsed storing vocabulary time: " << chrono.get_Seconds(5) << " seconds." << std:
    }
}
```

En esta función perteneciente al programa main, vamos en primer lugar a llevar a cabo el preprocesado de los datos y a continuación generamos el vocabulario y lo guardamos en el fichero correspondiente.

Gabriel Melián Hernández
alu0100819786
alu0100819786@ull.edu.es



La clase en sí contará con diversos atributos como son los ficheros de entrada y salida, el contador de cuántas palabras hay, el número de palabras diferentes que hay, el vocabulario en sí o la probabilidad de clase de cada palabra. Cada uno de estos atributos tiene sus métodos Getters y Setters correspondientes para su correcta manipulación.

Gabriel Melián Hernández
alu0100819786
alu0100819786@ull.edu.es

```
private:
    // Atributos.
    std::string inputFile_;         // Fichero de entrada.
    std::string outpuFile_;         // Fichero de salida donde será guardado el vocabulario.
    int vocabularyCounter_;         // Numero de diferentes palabras en el vocabulario.
    int nTokens_;                   // Número de palabras en el texto.
    int nLines_;                    // Numero de líneas leído
    std::set<Token> vocabulary_;    // Set con todo el vocabulario.
    std::string type_;              // El tipo del vocabulario.
    float classProbability_;        // La probabilidad de clase

public:
    // Constructores y Destructor.
    Vocabulary (void);
    Vocabulary (std::string inputFile, std::string outpuFile);
    ~Vocabulary (void);

    // Getters & Setters
    std::string get_InputFile (void) const;
    std::string get_OutpuFile (void) const;
    int get_VocabularyCounter (void) const;
    int get_NTokens (void) const;
    int get_NLines (void) const;
    std::set<Token> get_Vocabulary (void) const;
    std::string get_Type (void) const;
    float get_ClassProbability (void) const;

    void set_InputFile (std::string newInputFile);
    void set_OutpuFile (std::string newOutpuFile);
    void set_VocabularyCounter (int newVocabularyCounter);
    void set_NTokens (int newNTokens);
    void set_NLines (int newNLines);
    void set_Vocabulary (std::set<Token> newVocabulary);
    void set_Type (std::string newType);
    void set_ClassProbability (float newClassProbability);
```

Además también disponemos de las funciones encargadas de llamar al preprocesado, cargar las StoPWords, calcular las probabilidad y obviamente, generar, leer y almacenar el vocabulario.

```
// Funciones.
void preProcessData (std::string& stopWordFile);
std::vector<std::string> loadStopWord (std::string& inputFile);
void generateVocabulary (std::string& inputFile, bool tokenized);
void calculateProbabilities (void);
void addClassProbability (int size);

// Escritura.
void readVocabulary (std::string& inputFile);
void storeVocabulary (std::string& outputFile);
void readLearnedData (std::string& inputFile);
```

Para llevar a cabo la generación de nuestro vocabulario vamos a ir leyendo el fichero de entrada y comprobando si la palabra leída aparece o no en la lista que ya disponemos siendo en caso negativo que la añadimos y si ya está añadida incrementamos en una unidad su tasa de aparición.

```cpp
/**
 * @brief      Genera el fichero de Vocabulario.
 *
 * @param      inputFile  El fichero de entrada.
 * @param[in]  tokenize   Si quieres que las palabras se tokenizen o no.
 */
void Vocabulary::generateVocabulary (std::string& inputFile, bool tokenize) {
    std::ifstream file(inputFile, std::ios::in);
    if (file.fail()) {
        std::cout << std::endl << "Error 404, generateVocabulary file not found. (" << inputFile << ")"
        exit(1);
    }
    set_NTokens(0);
    set_VocabularyCounter(0);
    std::string word;
    std::set<Token>::iterator it;
    while (!file.eof()) {
        file >> word;
        if (isdigit(word[0])) {
            nLines_ = stoi(word);
        }
        else {
            if (!vocabulary_.count(word)) {
                Token newToken(word);
                vocabulary_.insert(newToken);
            }
            else if (tokenize){
                it = vocabulary_.find(word);
                Token newToken = *it;
                newToken.incrementate();
                vocabulary_.erase(word);
                vocabulary_.insert(newToken);
            }
            nTokens_++;
        }
    }
    file.close();
    set_VocabularyCounter(vocabulary_.size());
}
```

1.5.- Clase Corpus.

El paso siguiente sería el de generar un corpus para cada una de las clases que tenemos disponibles que son, Negative y Positive, para ello haremos uso de la clase Corpus, que al igual que la anterior dispone de una función propia en el main para llamarla y llevar a cabo la creación de nuestro corpus, que se almacenarán en un fichero aparte para cada una de las diferentes clases, almacenando el número de veces que se da cada clase y las palabras que forman su token propio.

```
/**
 * @brief      Genera un corpus para cada tipo de dato recibido como argumento por consola.
 *             El tipo de dato, tiene que ser la primera columna del fichero .csv seguido
 *             por una coma en el fichero de datos, pero no en la línea de comandos.
 *
 * @param      argc  La cantidad de argumentos.
 * @param      argv  El array de argumentos.
 */
void generateCorpus (int& argc, char* argv[]) {
    if (argc < 4) {
        std::cout << std::endl << "Error, the program needs at least 3 arguments to generate the corpus:"
        std::cout << std::endl << "Each \"CORPUS\" represents one data type that wants to be separated int
        exit(1);
    }
    std::string originFile = argv[2];
    std::string reservedWords = argv[3];
    PreProcesser preProcesser;
    Vocabulary voc;
    std::vector<std::string> stopWords = voc.loadStopWord(reservedWords);
    for (int i = 4; i < argc; i++) {
        std::string tmp = argv[i];
        Corpus newCorpus(tmp, originFile);
        newCorpus.generateCorpus(stopWords, preProcesser);
    }
}
```

Esta clase al igual que las demás tiene sus propios métodos Setters y Getters para trabajar con los atributos de cada clase, pero la función más importante es GenerateCorpus, que recibirá el procesado y las stopWords para llevar a cabo la generación de los corpus de cada uno de los tipos de clase que disponemos. Cargará las filas de datos donde el tipo elegido sea el correcto y les aplicará las medidas de preprocesados necesarias.

```
/**
 * @brief      Genera y almacena los datos del corpus.
 *
 * @param      stopWords     Las StopWords.
 * @param[in]  preProcesser  El preprocesado.
 */
void Corpus::generateCorpus (std::vector<std::string>& stopWords, PreProcesser& preProcesser) {
    Chrono myChrono;
    int dataLines;
    myChrono.startChrono();
    std::string outputFile = "../outputs/preProcesserHelper.txt";
    std::string dataType = get_Name() + ",";
    dataLines = preProcesser.loadData(inputFile_, dataType);
    preProcesser.convertLowerCase();
    preProcesser.erasePunctuationSigns();
    preProcesser.eraseAllNumbers();
    preProcesser.storeData(outputFile, 0);
    preProcesser.eraseReservedWords(stopWords, outputFile);
    preProcesser.storeData(outputFileName_, dataLines);
    myChrono.stopChrono();
    std::cout << std::endl << "Elapsed time for corpus " << name_ << ": " << myChrono.get_Seconds(5) << " seconds."
}
```

Gabriel Melián Hernández
alu0100819786
alu0100819786@ull.edu.es

```
/**
 * @brief      Esta clase describe un corpus.
 */
class Corpus {

    private:
        // Atributos
        std::string name_;              // El nombre del corpus o tipo de dato
        std::string outputFileName_;    // El nombre del fichero de salida.
        std::string inputFile_;         // El nombre del fichero de entrada.

    public:
        // Constructores y Destructor
        Corpus (void);
        Corpus (std::string name, std::string inputFile);
        ~Corpus (void);

        // Setters y Getters
        std::string get_Name (void) const;
        std::string get_OutputFileName (void) const;
        std::string get_InputFile (void) const;

        void set_Name (std::string newName);
        void set_OutputFileName (std::string newOutputFileName);
        void set_InputFile (std::string newInputFile);

        // Funciones
        void generateCorpus (std::vector<std::string>& stopWords, PreProcesser& preProcesser);
```

1.6.- Clase Learner.

La clase learner también dispone de una función en el main y su principal objetivo es el de generar una probabilidad de aparición para cada token disponible en cada uno de los corpus generados para nuestras clases.

```
/**
 * @brief      Calcula la probabilidad de cada token en el corpus pro
 *             como entrada y las guarda en otro fichero.
 *
 * @param      argc  La cantidad de argumentos.
 * @param      argv  El array de argumentos.
 */
void generateLearner (int& argc, char* argv[]) {
    if (argc < 3) {
        std::cout << std::endl << "Error, the program needs at least 
        std::cout << std::endl << "Each \"Data\" represents one data 
        exit(1);
    }
    Learner learner(argv, argc);
}
```

Gabriel Melián Hernández
alu0100819786
alu0100819786@ull.edu.es

Esta, es una clase que al igual que la anterior de Corpus, tiene los métodos de Setters y Getters y solo una función más pero es la de vital importancia ya que es la que va a generar nuestro fichero de aprendizaje con las probabilidades de cada uno de los tokens presentes en el corpus de la clase usando la función generateLogProb que habíamos visto con anterioridad.

```
/**
 * @brief      Genera las probabilidades para todos los tokens del vocabulario
 *             y los almacena en un fichero.
 */
void Learner::learnAndStoreData (void) {
    for (unsigned i = 0; i < learners_.size(); i++) {
        Chrono chrono;
        chrono.startChrono();
        std::string fileName = "../outputs/aprendizaje";
        fileName += inputCorpusFiles_[i][0];
        fileName += ".txt";
        std::fstream file(fileName, std::ios::out);
        if (file.fail()) {
            std::cout << "Error while storing learned data \"" << fileName << "\" is not valid document" <
            exit(1);
        }
        else {
            int tokenAmmount = learners_[i].get_NTokens();
            int vocSize = learners_[i].get_VocabularyCounter();
            std::string data = "Numero de documentos del corpus: " + std::to_string(learners_[i].get_NLine
            data += "\nNumero de palabras del corpus: " + std::to_string(tokenAmmount);
            for (auto tmp : learners_[i].get_Vocabulary()) {
                std::string line =  "\nPalabra: " + tmp.get_Name();
                data += line + " Frec: " + std::to_string(tmp.get_Ammount()) + " LogProb: " + std::to_stri
            }
            file << data;
            file.close();
        }
        chrono.stopChrono();
        std::cout << std::endl << "Elapsed time for calculating probabilities: " << chrono.get_Seconds(5)
    }
}
```

Gabriel Melián Hernández
alu0100819786
alu0100819786@ull.edu.es

```
/**
 * @brief      Esta clase describe un learner.
 */
class Learner {

    private:
        // Atributos.
        std::vector<Vocabulary> learners_;          // El learner para cada tipo de dato.
        std::vector<std::string> inputCorpusFiles_; // El fichero de entrada del corpus.
        std::string vocabularyFile_;                // El fichero del vocabulario.

    public:
        // Constructores y Destructor
        Learner (void);
        Learner (char* argv[], int& argc);
        ~Learner (void);

        // Setters y Getters
        std::vector<Vocabulary> get_Learners (void) const;
        std::vector<std::string> get_InputCorpusFiles (void) const;
        std::string get_VocabularyFile (void) const;

        void set_Learners (std::vector<Vocabulary> newLearners);
        void set_InputCorpusFiles (std::vector<std::string> newInputCorpusFiles);
        void set_VocabularyFile (std::string newVocabularyFile);

        // Funciones.
        void learnAndStoreData (void);
```

```
Numero de documentos del corpus: 18046
Numero de palabras del corpus: 393648
Palabra: <UNK> Frec: 0 LogProb: -13.000900
Palabra: aa Frec: 1 LogProb: -12.307754
Palabra: aaa Frec: 1 LogProb: -12.307754
Palabra: aaaaakubosan Frec: 1 LogProb: -12.307754
Palabra: aaaand Frec: 0 LogProb: -13.000900
Palabra: aaaas Frec: 0 LogProb: -13.000900
Palabra: aaan Frec: 0 LogProb: -13.000900
Palabra: aaanews Frec: 0 LogProb: -13.000900
Palabra: aaannnddd Frec: 0 LogProb: -13.000900
Palabra: aaanortheast Frec: 1 LogProb: -12.307754
Palabra: aabutan Frec: 1 LogProb: -12.307754
Palabra: aacopd Frec: 1 LogProb: -12.307754
Palabra: aacounty Frec: 0 LogProb: -13.000900
Palabra: aadeshrawal Frec: 0 LogProb: -13.000900
Palabra: aahealth Frec: 1 LogProb: -12.307754
```

Gabriel Melián Hernández
alu0100819786
alu0100819786@ull.edu.es

1.7.- Clase Classifier.

Por último tenemos nuestro clasificador que será el encargado de decir a qué clase pertenece un tweet dado gracias a las palabras que lo componen. Tiene una función propia en el main que lo llama.

```
/**
 * @brief      Clasifica el corpus de testeo.
 *
 * @param      argc  La cantidad de argumentos.
 * @param      argv  El array de argumentos.
 */
void generateClassifier (int& argc, char* argv[]) {
    if (argc < 4) {
        std::cout << std::endl << "Error, the program needs at least 3 arguments to classi
        std::cout << std::endl << "Each \"LEARNED\" is generated with the --learner flag."
        exit(1);
    }
    Classifier newClassifier(argv, argc);
}
```

Los Métodos Setters y Getters para los atributos de los ficheros de entrada, las clases, los datos, el resultado del aprendizaje y el resumen de salida y por supuesto, las funciones necesarias para llevar a cabo la clasificación tweet a tweet del fichero de testeo proporcionado y para almacenar los resultados

Gabriel Melián Hernández
alu0100819786
alu0100819786@ull.edu.es

```
class Classifier {

    private:
        // Atributos.
        std::vector<std::string> inputFiles_;   // Vector de fichero de entrada.
        std::vector<Vocabulary> class_;         // Vector de la Clase/Vocabulario.
        std::string data_;                      // Aqui almacenaremos los datos de salida
        std::set<Token> learnedData_;           // Diccionario con todo el aprendizaje.
        std::vector<std::string> resume_;       // Vector donde será almacenado el resume

    public:
        // Cnstructores y Destrcutor.
        Classifier (void);
        Classifier (char* argv[], int& argc);
        ~Classifier (void);

        // Setters y Getters.
        std::vector<std::string> get_InputFiles (void) const;
        std::vector<Vocabulary> get_Class (void) const;
        std::string get_Data (void) const;

        void set_InputFiles (std::vector<std::string> newInputFiles);
        void set_Class (std::vector<Vocabulary> newClass);
        void set_Data (std::string newData);

        // Funciones.
        void classifyFile (std::string& inputFile, std::string& stopWords);
        void classify (std::vector<std::string> sentence);
        void generateClassProbability (void);
        void preProcess (std::vector<std::string>& stopWords, std::string& sentence);

        // Lectura y Escritura.
        void readInputFiles (char* argv[], int& argc);
        void storeFile (std::string& outputFile, std::string& resumeFile);
};
```

Para llevar a cabo la clasificación de los tweets usaremos la función “Classify” que recibirá el tweet e irá buscando las probabilidades de cada una de las palabras que lo componen hasta conseguir el total y elegir entre ambas clases la que sea más probable que pertenezca al tweet y lo clasificamos en dicha clase.

Gabriel Melián Hernández
alu0100819786
alu0100819786@ull.edu.es

```cpp
/**
 * @brief      Clasifica un tweet.
 *
 * @param[in]  sentence  El tweet que va a ser clasificado.
 */
void Classifier::classify (std::vector<std::string> sentence) {
    std::vector<float> prob;
    prob.resize(inputFiles_.size());
    for (unsigned i = 0; i < prob.size(); i++) {
        prob[i] = 0.0;
    }
    std::set<Token>::iterator it;
    for (unsigned i = 0; i < sentence.size(); i++) {
        it = learnedData_.find(sentence[i]);
        for (unsigned j = 0; j < prob.size(); j++) {
            prob[j] += it -> get_MultiClass(j);
        }
    }
    data_ += ", ";
    unsigned selection = 0;
    for (unsigned i = 0; i < prob.size(); i++) {
        prob[i] += class_[i].get_ClassProbability();
        if (prob[selection] < prob[i]) {
            selection = i;
        }
        data_ += std::to_string(prob[i]);
        data_ += ", ";
    }
    resume_.push_back(class_[selection].get_Type());
    data_ += class_[selection].get_Type();
    data_ += ".\r";
}
```

Text-Classifier-master > outputs > clasificacion_alu0100819786.csv

```
1  nce #restezchezvous #StayAtHome #confinement https://t.co/usmuaLq72n N, -195.95, -207.04, N.
2  e to fight against COVID 19?. #govindia #IndiaFightsCorona N, -195.67, -199.04, N.
3  erment  is guilty of  being  irosponcible  with life  on a global scale N, -229.58, -244.21, N.
4  left to checkout staff to police the actions of the selfish and profiteer N, -178.52, -190.47, N.
5  ted new protocols due to the COVID-19 coronavirus. https://t.co/5CecYtLnYn N, -151.98, -161.49, N.
6  l keep you updated on how I'm doing ???? No panic. https://t.co/Lg7HVMZglZ N, -191.38, -194.04, N.
```

Gabriel Melián Hernández
alu0100819786
alu0100819786@ull.edu.es

1.8.- Calcular El Error.

Por último una vez conseguido nuestro clasificador vamos a calcular el porcentaje de acierto y error a la hora de clasificar nuestros tweets. Vamos a hacer uso del conjunto de entrenamiento proporcionado y compuesto por 33444 tweets. Para ello vamos a utilizar el fichero COV-test.csv donde tenemos cada uno de los tweets pero sin la clase y el fichero resumeExpected.csv que contendrá el resultado esperado.

Una vez tengamos realizado nuestra clasificación, en el main, tendremos una función que llevará a cabo el calculo del error.

```cpp
/**
 * @brief      Calcula el porcentaje de error y acierto comparando
 *             los resultados generados y los esperados.
 *
 * @param      argc  La cantidad de argumentos.
 * @param      argv  El array de argumentos.
 */
void calculateError (int& argc, char* argv[]) {
    if (argc != 4) {
        std::cout << std::endl << "Error, the program needs 4 arguments to calculate the success
        exit(1);
    }
    std::string expected = argv[3];
    std::string received = argv[2];
    std::vector<std::string> expect;
    std::vector<std::string> receive;
    std::string tmp = "";
    std::ifstream file(expected, std::ios::in);
    if (file.fail()) {
        std::cout << std::endl << "Error 404," << expected << " file not found." << std::endl;
        exit(1);
    }
    else {
        while (!file.eof()) {
            file >> tmp;
            expect.push_back(tmp);
        }
        file.close();
    }
    std::ifstream file2(received, std::ios::in);
    if (file2.fail()) {
        std::cout << std::endl << "Error 404," << received << " file not found." << std::endl;
        exit(1);
    }
```

Gabriel Melián Hernández
alu0100819786
alu0100819786@ull.edu.es

```
else {
    file2 >> tmp;
    while (!file2.eof()) {
        file2 >> tmp;
        receive.push_back(tmp);
    }
    file2.close();
}

int size = 0;
if (expect.size() < receive.size()) {
    size = expect.size();
}
else {
    size = receive.size();
}
int correct = 0;
for (int i = 0; i < size; i++) {
    if (expect[i] == receive[i]) {
        correct++;
    }
}
float percentage = correct;
percentage /= size;
percentage *= 100;
std::cout << std::endl << "Success percentage: " << correct << " / " << size << " = " <<
std::cout << std::endl << "Error percentage: " << (size - correct) << " / " << size << "
```

que en nuestro caso tras llevar todo el proceso a cabo, es el siguiente:

```
make runTestPercentage
make[1]: Entering directory '/workspace/Cosas/Text-Classifier-master/build'
../bin/textClassifier --error ../outputs/resumen_alu0100819786.csv ../inputs/resumeExpected.csv

Success percentage: 33258 / 33444 = 99.4438 %.
Error percentage: 186 / 33444 = 0.556152 %.

Program finished correclty.
make[1]: Leaving directory '/workspace/Cosas/Text-Classifier-master/build'
gitpod /workspace/Cosas/Text-Classifier-master/build (main) $
```

Hemos conseguido un 99,44% de acierto a la hora de clasificar los tweets con el fichero de entrenamiento.

Gabriel Melián Hernández
alu0100819786
alu0100819786@ull.edu.es