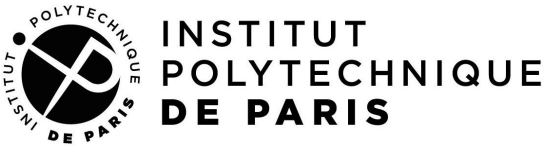

Categorization and extraction of severe weather events of European Severe Weather Database

A PRL defense submitted for
Bachelor in Computer Science

by

Mija Pilkaite

at the

Computer Science, DaSciM group, LIX
Ecole Polytechnique
France

under the supervision of

Davide Buscaldi (LIX)

Contents

Abstract

Amidst the concerns for climate change, severe weather events represent an important issue because of their impact on human society [5]. Researchers have been collecting data regarding these kinds of events to try to understand their relationship to climate change and improve our ability to predict and prepare for them. The European Severe Storms Laboratory has created the European Severe Weather Database (ESWD) [1], which allows the public to report and share information about severe weather events. However, the reliance on crowd-sourced observations can lead to a bias towards areas with higher population densities [2] which can be problematic for accurately understanding the distribution of severe weather events.

In today's world, climate change and the dangers inflicted on society by it are very common concerns. To address this issue, there has been growing interest in using automatic processing of various media sources, such as social media, to identify and survey severe weather events more accurately and objectively. This approach has been shown to be effective in the context of events such as floods [3, 4], and can provide important information for understanding the distribution and impact of severe weather events.

Thus, this project focuses on using different techniques to extract the relevant information and make the ESWD less human-dependent.

Chapter 1

Introduction

In this chapter, we will give a short overview of the main aspects of this research project. We will first introduce the goal and idea behind the European Severe Weather Database and the need for this type of research regarding it. We will look into a very general problem of the classification of different severe weather events and the classification between references about actual severe weather events and false data. We will discuss the models and techniques used and analyze which ones yielded the best results.

1.1 Introduction of the project

Firstly, as the advancements in the Machine Learning world have been fast-paced, we are looking for more adaptations and uses for it in the science world. Climate change has been also a burning topic for several years, thus this project provides an opportunity of connecting these two fields in order to better explore the severe weather events in Europe. The analysis and prediction of severe weather events are of great importance, as extreme weather conditions can lead to significant economic and human losses [5] and accurate categorization and extraction of severe weather events can help understand the frequency, intensity, and distribution of such disasters, as well as for planning and designing effective mitigation strategies [6]. The European Severe Weather Database (ESWD) serves as a valuable resource for weather data, containing reports of severe weather events in Europe within most of the countries and a wide range of categories [1]. The ESSL started as an informal network of European scientists to advance research on severe convective storms and extreme weather events on a European level and today it is an actively growing database that could highly benefit from a method, that would help attain data easier and remove or at least minimize human power and error from the maintenance of the database. Thus, in this case, we are trying to build as accurate language processing models as possible to correctly classify text extracts within severe weather categories and differentiate between false and true data (whether the event is a severe weather event or

not).

When it comes to the project, there were multiple characteristics of the data set that could be interesting in this discussion. To begin with, the data set used contained both English and French severe weather extracts. Thus, separate processes had to be done in both languages. Some of the references were noisy and did not contain relevant information, thus making the models somewhat less precise. Moreover, references were provided for about 3000 entries out of the 18000 thus making the data set significantly smaller.

Data were acquired from the European Severe Weather Database and only covers the region of France, thus the results are influenced by the most common weather events in France.

Finally, as the references were rather noisy, this research is restricted by only extracting the type of event from the given text and not more information.

1.2 Objectives and Methodology

In this project, our main objectives are to classify severe weather events from text using the ESWD categorization and to classify data that is specifically collected to be falsely recognized (such as descriptions containing the most common words or descriptions of not severe weather events).

The project focuses on the utilization of different feature representation methods and various machine learning models with the goal of discovering the best combination that could accurately predict a type of event even from noisy data. Our objectives for this project were:

1. To investigate and get to know different feature representation methods, such as Bag of Words, Term Frequency-Inverse Document Frequency (TF-IDF), and Sentence Bert (SBert), in order to accurately represent the data within the references and evaluate performances in two different languages - English and French.

2. To evaluate different models - logistic regression, random forest, and fully connected neural network, for each of the feature representations to notice which combinations work together the best.

3. To extract the most important words from the database in order to be able to create a fitting second database for the second part of the project.

4. To develop a binary classifier capable of differentiating between false and true severe weather events even if both references contain keywords.

As mentioned above, the methodology used consists of:

1. First, some of the data is removed to get rid of the null values and the references that do not contain relevant information. Then, we preprocess the data using Spacy library, where the dataset is first split into English and French using a Spacy Language Detector. Then, tokenization, lemmatization, removing stopwords, and punctuation are performed in order to prepare the references for further analysis.

2. Then, different techniques of feature representation are performed. The functions are done by hand in order to get a better grasp of the techniques behind them.

3. The dataset is then split into training and testing sets. The models used are written and adapted to the different feature representations.

4. Feature selection is conducted to find the most important words.

5. Accuracy and error metrics are analyzed. The confusion matrix is done in order to see which groups get misclassified the most.

6. A binary classifier is built to differentiate between false and true severe weather events using Logistic Regression and Random Forest models with a Bag of Words and TF-IDF features.

Chapter 2

Classification of Severe Weather Events

2.1 Data Analysis and Preprocessing

When first discovering a dataset, it is crucial to get to know the data, how it is distributed, and what features are represented. To begin with the project, the decision was taken to only use the 'TYPE EVENT' column and 'REFERENCE' column. These two were the main data sources needed to build a classifier for the weather event types. As we can see, our dataset consists of 8 different severe weather events and is highly imbalanced.

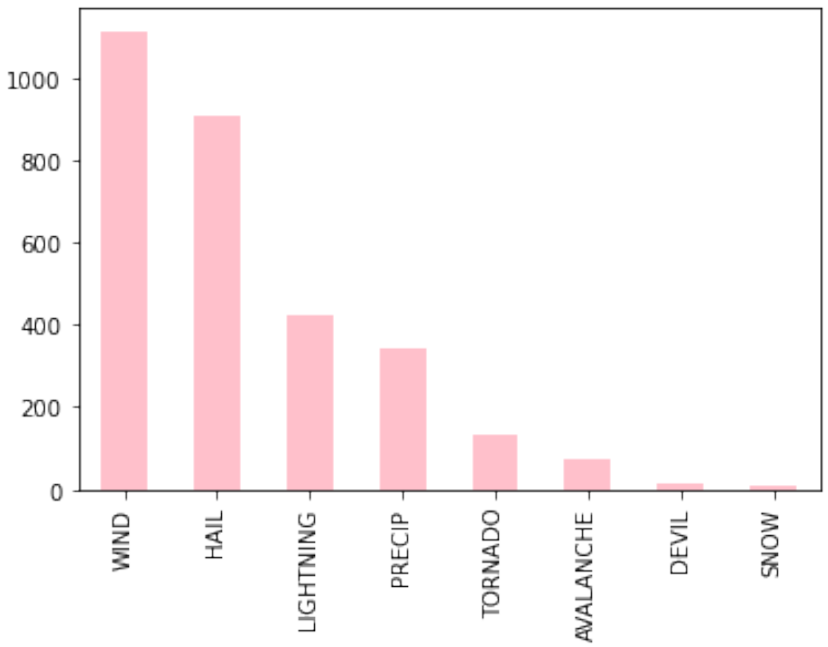

Figure 2.1: As it is seen in this figure, we have 8 unique event types and a big disbalance between them.

Let us explore the frequency of each event quantitatively.

	TYPE_EVENT
WIND	16352
HAIL	3064
PRECIP	1845
LIGHTNING	553
TORNADO	291
AVALANCHE	109
DEVIL	65
SNOW	38

Figure 2.2: Frequency of events in the dataset

Then, moving on to data preprocessing we will take several steps to ensure that our data is ready to be preprocessed. This step betters the performance of our models built later as we simplify the process of feature representation. Below, we will discuss steps taken in this part of our project - language separation, tokenization, lemmatization, and removal of punctuation and stopwords.

The first obstacle found was that the ESWD database provides references in two different languages - English and French. As the preprocessing process is different for these languages is different and thus the dataset had to be separated. To achieve this, we utilized the Python library SpaCy, which provides a powerful and efficient language detection module [7]. By applying SpaCy's language detection capabilities to each event description, we were able to separate the dataset into distinct English and French subsets, which could then be preprocessed and analyzed separately. At the end of this step, we ended up with 424 English entries and 2526 French entries. However, after further exploration, it was noticed that the English dataset contained mostly noise and references that did not contain sufficient information to be classified. First, some cleaning was done in order to remove the empty references, such as these ones:

"Joel Feneule (on Facebook), 17 Nov 2022."
"Report via Kachelmannwetter.com, 4 Nov 2022."
"Eyewitness report via La Météo de la Somme (on Facebook), 23 OCT 2022."

All the references containing words "Facebook" or "Twitter" and that are shorter than 8 words were removed, as we also have some references as these:

"Eyewitness report via Alpes 1 (on Facebook), 14 SEP 2022. ""Hautes-Alpes : un violent orage de grêle s'est abattu sur Gap"

However, even then most of the data was either noise or some French data, that was unrecognized during the language separation. Thus, from this point onwards, we decided to only move forward with the French part of the dataset.

As a next step, we moved on to tokenization, which is the process of breaking down a text into individual words or tokens. Again, using the SpaCy library for this step, we converted our sentences into lists of tokens for easier creation of vectors.

Afterward, the lemmatization step reduced the words to their base case (also known as lemma). This helps us convert different tenses and conjugations of verbs, and plurals of nouns be converted in a single representation. This step helps us reduce the dimensionality of the text data and thus have smaller vectors for the representation of the sentences. Applying SpaCy's lemmatization functionality to the tokenized event descriptions, we converted the tokens into their respective lemmas, further refining the text data.

Finally, the last step in data preprocessing was removing stopwords and punctuation. Punctuation marks, such as commas and points, usually do not carry a lot of meaning in the sentence, thus just making our data messier. The same applies to stopwords (such as "il", "elle", "et"), which are common words in the language but do not give away much context by themselves. This helps us again reduce the dimensionality of the data and create shorter vectors for the representation of our references. For stopwords, we employed SpaCy's built-in lists of French stopwords to identify and remove them from the tokenized and lemmatized event descriptions.

By applying these techniques to our data, we transformed the raw ESWD text data into a clean and structured format, suitable for further feature extraction and classification tasks. This will be useful for our further feature representation techniques.

2.2 Models

In this section, we will thoroughly look through the models used on all of the feature representations to see which performs best.

2.2.1 Logistic Regression

For our project logistic regression is a powerful method to do multi-class classification by estimating probabilities (between our references and possible severe events) using a logistic function [8] Logistic regression, as seen in the course, uses the same idea as linear regression but applies a sigmoid function on the output to retrieve the result that is a probability from 0 to 1. [9] The non-linear function that is applied, otherwise known as sigmoid σ function, is known as:

$$\sigma(a) = \frac{1}{1 + exp(-a)}$$

where for any $a \in (-\infty, +\infty)$ it will squish it so that $a \in [0, 1]$ [10]

If we want to do a multiclass classification, we can use multiclass logistic (also known

as softmax) regression. In this case, the model estimates the probability of that reference belonging to each class. [11]. The softmax function is used to convert the linear combination into probabilities for each category. The softmax function for class i is defined as:

$$\text{softmax}(z_i) = \frac{exp(z_i)}{\sum exp(z_j)}$$

where z_i is the linear combination for class i, and the sum is taken over all possible classes.

The goal of logistic regression is to find the best parameters for the problem (weights and bias) that minimize the negative log-likelihood (or log-loss) of the model [8]. Thus, this technique allows us to use logistic regression to predict the type of event from the reference. We configure the model using the Newton-SG solver and set the multiclass parameter to multinomial in the function from the SKlearn library. We chose this solver as Newton-CG (Conjugate Gradient) solver is a variant of the Newton-Raphson method that is well-suited for large-scale logistic regression problems [12] It has several advantages over other techniques, such as faster convergence (as this solver uses second-order derivatives retrieved from the Hessian matrix with the gradient descent information), it is more robust than a simple gradient descent and it does not require tweaking the learning rate. It requires more computational power, as one has to invert and compute the Hessian matrix, however as our dataset is quite small, it is a good fit. [12]

Thus, we will use this logistic regression model with all three of the feature representations.

2.2.2 Random Forest

The next model we chose to use was a Random Forest, which is an ensemble (a technique that combines multiple base models to create a more accurate and robust final model) learning method, where multiple decision trees are constructed and their results are combined to improve accuracy and reduce overfitting. [13] The diversity between multiple decision trees is retrieved, as each is trained on a random subset of the training data, and a random subset of features is considered for splitting at each node, thus the trees are different from one another. Each decision tree in the ensemble is a binary tree structure where nodes represent feature tests and leaf nodes represent class predictions. The goal of each tree is to recursively split the data based on the feature that maximizes the information gain [14]. Each tree in the random forest gives a prediction and in the classification tasks, the most common one wins.

Random Forest is a robust and more precise technique than a simple decision tree, as the risk of overfitting is reduced. This technique is also good for handling noisy, imbalanced, and high-dimensional data, which is the case in our dataset. In order to find the best parameters for our dataset, we employed hyperparameter tuning. It was performed on the Random Forest classifier using GridSearchCV. The hyperparameters

tuned were:

n estimators: The number of trees in the forest. Tested values were [10, 50, 100, 200].

max depth: The maximum depth of the tree. Tested values were [None, 10, 20, 30, 40]. A value of None indicates no maximum depth.

min samples split: The minimum number of samples required to split an internal node. Tested values were [2, 5, 10].

min samples leaf: The minimum number of samples required to be at a leaf node. Tested values were [1, 2, 4].

GridSearchCV performed an exhaustive search over the parameter grid, fitting 3-fold cross-validation on each parameter combination. A total of 180 parameter combinations were tested, resulting in 540 fit Random Forest models.

2.2.3 Fully Connected Neural Network

For our last model, we chose to implement a fully connected neural network. In our model, we have each neuron in one layer connected to every neuron in the adjacent layers. A feedforward deep learning model capable of learning complex patterns and relationships in the data is a good choice for the ESWD classification project. Let us analyze the architecture of this neural network model.

The input layer is the starting point of any neural network. It takes the input features, which are different feature representations described below and passes them to the next layer. The input shape is that of a vector, which is the size of the vocabulary. The first dense layer serves as a hidden layer in the network. The choice of using a dense layer is logical because it allows the network to learn complex, non-linear relationships between input features and the target variable. The Rectified Linear Unit (ReLU) activation function is chosen because it is computationally efficient, introduces non-linearity, and helps mitigate the vanishing gradient problem. Dropout layer: The dropout layer is added to improve the network's generalization and prevent overfitting. During training, a fraction of neurons (20 percent in this case) is randomly shut down, along with their connections. This forces the network to learn redundant representations, which in turn helps improve generalization performance on unseen data. Second Dense layer with ReLU activation: Another dense layer is added to increase the capacity of the model, allowing it to learn more complex patterns in the data. Like the first dense layer, this layer also uses the ReLU activation function for the same reasons mentioned earlier. Dropout layer: Another dropout layer is added after the second dense layer to further improve generalization and prevent overfitting. Output layer with Softmax activation: The output layer is a dense layer with as many units as the number of unique classes in the target variable. The choice of the Softmax activation function is logical because it is specifically designed for multiclass classification tasks. Softmax converts the linear combinations

into probabilities that sum up to one across all classes, making it easier to interpret the predictions.

Algorithm 1 Multiclass Classification Neural Network

1: **Input:** Training dataset X_{train}, y_{train} and test dataset X_{test}, y_{test}
2: Encode labels in y_{train} and y_{test} using LabelEncoder
3: One-hot encode the encoded labels
4: Initialize Sequential neural network model
5: Add Dense layer with 512 units, ReLU activation, and input shape
6: Add Dropout layer with dropout rate 0.2
7: Add Dense layer with 512 units and ReLU activation
8: Add Dropout layer with dropout rate 0.2
9: Add Dense output layer with Softmax activation and units equal to the number of unique classes
10: Compile the model with categorical cross-entropy loss, Adam optimizer, and accuracy metric
11: Train the model on X_{train} and y_{train} one-hot encoded, with batch size 128 and 20 epochs
12: Validate the model on X_{test} and y_{test} one-hot encoded
13: Evaluate the model and print test loss and accuracy
14: **Output:** Trained neural network model

This algorithm is capable of learning complex data relationships thus making it a good model for our multiclass classification. However, neural networks do have some downsides to consider such as complexity, as they are hard on the computer and hard to understand, they also tend to overfit if there is not enough data. We will analyze its performance in the next sections.

2.3 Feature Represenation techniques

2.3.1 Bag of Words

The Bag of Words (BoW) model is a widely used text representation technique in the field of natural language processing (NLP) and information retrieval (IR). It represents a document as an unordered set of words, disregarding grammar and word order but keeping track of the frequency of each word occurrence [15]. The BoW model can provide a simple yet effective approach to extracting meaningful features from unstructured data, such as text documents or, in the case of the ESWD project, textual descriptions of severe weather events.

The Bag of Words method is based on the idea that the frequency of the word within a document is correlated with its importance and can give us information about the content even if the order of words is not kept the same [16] Thus, we have two main components in this feature representation method, which are a dictionary and a representation vector.

First, we create a dictionary in which every unique word in the dataset is included and it serves as a reference for building the representation vectors. Then, for each entry in a dataset, we construct a vector by counting the frequency of each word in the dictionary. This means that every representation vector is the same length (length of a dictionary of a dataset) and for its value, we have the frequency of that word in this specific reference. Mathematically, let D be the dictionary and r be a reference. The feature vector $F(r)$ is given by:

$$F(r) = [f(w_1, r), f(w_2, r), ..., f(w_n, r)]$$

where where $f(w_i, r)$ is the frequency of word w_i in reference r.

We choose the Bag of Words method for this project because of its three main traits. Simplicity: The BoW model is relatively simple to understand and implement, thus making a great first step in severe weather classification. Scalability: The BoW model can handle large datasets efficiently and can be easily parallelized for even faster processing [17] Extensibility: This representation can be further refined using the TF-IDF vector, which will be discussed in the next section, or other techniques such as latent semantic indexing (LSI) [18]

After applying the functions to our English and French datasets, we get vectors of size 88 for the English dataset and 96 for the French dataset.

2.3.2 TF-IDF vector

In this subsection, we focus on using the Term Frequency-Inverse Document Frequency (TF-IDF) method to extract relevant information from textual data and incorporate it into our project in order to better understand and interpret the references.

This method is a numerical statistic that reflects the importance of a word in a document relative to a collection of documents (corpus). It is widely used in text mining and information retrieval tasks. The mathematical formulation of TF-IDF consists of two main components: Term Frequency (TF) and Inverse Document Frequency (IDF) [19]. It has been widely used in text mining and information retrieval tasks for quite some time and should bring better accuracy than Bag of Words representation.

The vector consists of two parts - Term Frequency (TF) and Inverse Document Frequency (IDF). Term Frequency (TF) measures the number of times a term (word) appears in a document. It is the simplest way to represent the importance of a term in a document. However, using raw term frequency can be misleading, as longer documents tend to have higher term frequencies. To account for this, we normalize the term frequency by dividing it by the total number of terms in the document [20] Mathematically, we obtain this by calculating:

$$TF(t, d) = \frac{(\text{Number of times term } t \text{ appears in document } d)}{(\text{Total number of terms in document } d)}$$

Inverse Document Frequency (IDF) is a measure of how important a term is across the entire corpus. It assigns more weight to rare terms and less weight to common terms that are present in many documents. The idea behind IDF is that terms that appear frequently in many documents might not be very informative or discriminative for specific documents [20]

Mathematically, the inverse document frequency for a term t in a corpus is:

$$IDF(t) = \log\left(\frac{N}{n_t}\right)$$

where:

1. N is the total number of documents in the corpus

2. n_t is the number of documents containing term t

Finally, the TF-IDF value for a term t in document d is calculated by multiplying the TF and IDF values:

$$TFIDF(t, d) = TF(t, d) * IDF(t)$$

The resulting TF-IDF value reflects the importance of a term in a specific document relative to the entire corpus [21]. High TF-IDF values indicate that a term is important within a specific document but relatively rare across the entire corpus. This makes TF-IDF a useful metric for tasks like text classification, document clustering, and information retrieval, as it helps to distinguish between different types of documents based on the relative importance of their terms.

The advantages of TF-IDF vector, especially when compared to Bag of Words method, the more significant words in the document have higher weights and it differentiates the words that are just frequent and truly significant. This technique is also easily comprehensible and works well with most of the machine learning models, thus making it a great choice for severe weather event classification. There are several drawbacks that come with this method, such as semantic understanding, where the context and the order of words are ignored, thus making the analysis less precise. This issue will be addressed with the next feature representation technique. TF-IDF tends to not perform very well on long documents and noisy data, which in our case is only a slight problem. References are short, however, reducing the noise in the references could be the key to higher performance accuracy.

Thus, TF-IDF vector is a good choice for our research and its performance with different models will be assessed in the next chapter.

2.3.3 SBert Transformation

The feature representations techniques above were rather common and did not capture the semantic meaning of a sentence, thus we turn to Sentence-BERT (SBERT), an advanced technique enabling better semantic understanding and multilingual sentence embeddings. Sentence-BERT (SBERT) modifies the BERT (Bidirectional Encoder Representations from Transformers) architecture, showing state-of-the-art performance in various natural language processing tasks [22]. SBERT addresses BERT's limitations for generating sentence embeddings [23]. While powerful, BERT is computationally expensive and not optimized for fixed-size sentence representations. SBERT overcomes these limitations using a Siamese network architecture. It takes two input sentences, processes them through separate BERT models with shared weights, and passes the outputs through a pooling layer generating fixed-size sentence embeddings. The Siamese architecture is trained using contrastive loss, minimizing the distance between semantically similar sentences and maximizing the distance between dissimilar ones.

In order to better grasp the logic behind Bert, let us take an example. Consider two sentences, A and B, with tokenized inputs, X_A and X_B. In a Siamese architecture, both sentences pass through separate BERT models with shared weights. Let C_A and C_B denote the contextualized representation outputs.

The pooling layer takes these contextualized representations and computes the sentence embeddings S_A and S_B, typically by taking the mean or maximum of the contextualized representations:

$$S_A = \text{pool}(C_A)$$

$$S_B = \text{pool}(C_B)$$

SBERT's training objective is minimizing the contrastive loss function, encouraging the model to generate embeddings closer for similar sentences and farther for dissimilar ones:

$$L(S_A, S_B, y) = (1 - y) \cdot \text{sim}(S_A, S_B) + y \cdot \max(0, m - \text{sim}(S_A, S_B))$$

Here, y is a binary label indicating whether two sentences are semantically similar, $\text{sim}(S_A, S_B)$ is a similarity function (e.g., cosine similarity), and m is a margin hyperparameter.

Using SBERT for Multilingual Classification of Severe Weather Events SBERT supports multilingual sentence embeddings using models trained on multilingual corpora, e.g. XLM-RoBERTa [24]. These models generate sentence embeddings for different languages in the same semantic space, suitable for cross-lingual applications.

For this project, we use a multilingual SBERT model to generate sentence embeddings

for French and English references. These embeddings are input features for a machine learning classifier to categorize the severe weather events the references describe. By leveraging SBERT's semantic understanding and cross-lingual capabilities, we can achieve improved classification performance over traditional text representation techniques.

Also, note that SBERT saves us from using language detection, which unfortunately is rather imprecise on short sentences and has a model, specifically built on working on multiple languages: "distiluse-base-multilingual-cased-v1". Thus, this makes it especially a good choice for our multilingual dataset. It also should address the limitations of BoW and TF-IDF methods and let us achieve a more accurate classification of severe weather events across languages.

2.4 Process and results

2.4.1 Bag of Words

We began with the Bag of Words feature representation on our French dataset. After converting to Bag of Words embeddings using the CountVectorizer from Sklearn we obtain dictionaries of length of 2918 for the French dataset. After splitting the data into train and test subsets (for the test subset we use 20 percent), we run the models described above.

With Logistic Regression we obtain an accuracy of 83.5% and F1 score of 83.3%. Let us take a look at the French dataset confusion matrix.

	A	D	H	L	P	S	T	W
AVALANCHE	12	0	0	0	1	0	0	0
DEVIL	0	0	1	0	1	0	0	0
HAIL	0	0	136	0	7	0	1	8
LIGHTNING	0	0	4	63	5	0	0	2
PRECIP	0	0	8	2	40	0	1	13
SNOW	0	0	0	0	0	0	0	1
TORNADO	0	0	3	0	2	0	9	6
WIND	0	0	9	1	10	0	0	160

Table 2.1: Confusion matrix for BoW for Logistic Regression

As we can see, most of the events are classified to their correct categories. Let us look at some of the misclassified examples:

Actual: LIGHTNING Predicted: WIND Reference: ['Orages', 'départ', 'feu', 'maison', 'détruite', 'arbre', 'couché', 'tarn', 'Haute-Garonne', 'tarn-et-garonne', 'FRANCE', 'TV', 'INFO', '30', 'aug', '2022']

Actual: HAIL Predicted: WIND Reference: ['violent', 'orage', 'évacuation', 'Saint-Etienne', 'femme', 'prisonnier', 'voiture', 'Villars', 'Progrès', '01', 'Jul', '2019']

Actual: LIGHTNING Predicted: HAIL Reference: ['intempérie', 'orage', 'faire', 'gros', 'dégât', 'ouest-aveyron', 'vendredi', 'soir', 'ladepeche.fr', '29', 'june', '2020']

Actual: HAIL Predicted: PRECIP Reference: ['meteo', 'cote', 'Azur', 'Direct', 'Facebook', '09', 'March', '2021']

Most of the misclassifications stem from words that can be used for multiple events. For example, the word 'orage' for the model is associated with the wind, but it can indicate also lightning and hail as we can see in the first two examples. Some of the other references simply do not contain the text we need to classify the events.

In order to determine the most salient features for a multiclass text classification task, we employ a feature weighting technique. Specifically, we train a logistic regression model with L1 regularization (Lasso) on the training data. The L1 regularization penalizes the absolute values of the feature weights, driving many weights to exactly 0 and effectively performing feature selection.

The features with non-zero weights after training are the most important for the classification task. We extract these feature weights from the trained logistic regression model. Then, for each target class, we sort the features by their weight magnitude in descending order. The top n features with the highest weights are selected as the most important features for that class. Now, let us take a look at the most important features:

	AVALANCHE	DEVIL	HAIL	LIGHTNING	PRECIP	SNOW	TORNADO	WIND
1	avalanche	tourbillon	grêle	foudre	inondation	neige	tornade	ws
2	skieur	jul	grêlon	incendie	orages	15	2019	meteofrance
3	avalanch	apr	limousin	foudroyer	inonder	jan	oct	arbre
4	jan	jardinier	jun	kachelmannwetter	boue	000	facebook	mini
5	savoie	ferté	agriculteur	jun	pluie	souffert	nov	feb
6	2021	bernard	eyewitness	sep	eau	le	dec	tempête
7	dec	surprendre	report	maison	provence	lorrain	direct	vent
8	alpes	ouest	centre	nord	oise	alsace	keraunos	coup
9	mort	tornad	bilan	feu	orage	chute	youtube	report
10	feb	argelès	21	com	progrès	priver	azur	march

Table 2.2: Most important features for logistic regression using BoW

As we can see, they are closely correlated to the events we are classifying and we can see how these keywords help the model decide what event are we referencing.

Now, to explore the Random Forest we follow the same procedure. We perform the hyperparameter tuning procedure described above and retrieve that the best parameters are:

1. 'max depth': None,
2. min samples leaf': 1,
3. 'min samples split': 5,
4. 'n estimators': 100

After running a Random Forest using these parameters, we get 84.3% accuracy and an F1 score of 83.4%. We check the confusion matrix to see more carefully what is happening

	AVALANCHE	DEVIL	HAIL	LIGHTNING	PRECIP	SNOW	TORNADO	WIND
AVALANCHE	14	0	0	0	0	0	0	0
DEVIL	0	0	0	0	0	0	0	2
HAIL	0	0	111	1	9	0	0	2
LIGHTNING	0	0	3	77	4	0	0	6
PRECIP	0	0	9	6	28	0	0	9
SNOW	0	0	0	0	0	0	0	1
TORNADO	0	0	1	0	1	0	5	5
WIND	0	0	6	6	5	0	0	161

Table 2.3: Confusion matrix for Random Forests using BoW

in the classification. As we can see, precipitation and wind are the events that are most likely to be misclassified.

Now, let us look more precisely at the examples that were misclassified:

Actual: WIND Predicted: PRECIP Reference: ['Orages', 'Saint-Etienne', 'Gier', '2618', 'impact', 'foudre', 'relever', 'progrès', '22', 'JUL', '2020']

Actual: PRECIP Predicted: HAIL Reference: ['météo', 'Saône-et-Loire', '71', 'v.', 'Twitter', '09', 'JUN', '2021', 'Panique', 'Autun', 'fort', 'averse', 'journal', 'SAÔNE-ET-LOIRE', '09', 'JUN', '2021']

Actual: WIND Predicted: HAIL Reference: ['Orages', 'grêle', 'faire', 'casse', 'auvergne', 'Rhône', 'Alpes', 'France', 'Info', '3', 'Jul', '2019']

As we can see again, keywords like already seen "Orages" tend to confuse the model and we again have some references that do not have sufficient information within. random

Rank	Feature
1	foudre
2	grêle
3	com
4	meteofrance
5	report
6	facebook
7	météo
8	ws
9	2022
10	eyewitness

Table 2.4: Most important features for Random Forest using Bag of Words

forest feature importance provides an aggregate measure of a feature's usefulness over all possible classes. While the scores confirm which features are most informative to the overall model, they do not specify precisely how or for which classes a given feature is most predictive. The random forest methodology limits feature importance to a single global ranking.

The top 10 features based on the Gini importance scores for our random forest model are presented in Table 2.4.1. These features represent words, phrases, and weather phenomena that are highly indicative of severe events based on their prominence in witness reports. Unfortunately, as we can see some of them do not indicate any weather events and are simply noise from the references. This could be the reason why our accuracy is not perfect.

Finally, let us explore the Fully Connected Neural Network we built (algorithm described above). Due to a lack of computational power, it was hard to tune the hyperparameters so we picked a guess that seemed satisfying. In order to achieve better accuracy we could find the parameters best fit for our model. However, we get a testing accuracy of 82.6%, which is similar to what we attained using other methods. As extraction of the most important features is quite complicated in neural networks, we will limit ourselves to exploring the confusion matrix and the incorrectly classified references.

	Predicted Class							
Actual Class	AVALANCHE	DEVIL	HAIL	LIGHTNING	PRECIP	SNOW	TORNADO	WIND
AVALANCHE	14	0	0	0	0	0	0	0
DEVIL	0	0	0	0	1	0	0	1
HAIL	0	0	109	2	10	0	2	0
LIGHTNING	0	0	4	75	7	0	0	4
PRECIP	0	0	9	5	24	0	0	14
SNOW	0	0	0	1	0	0	0	0
TORNADO	0	0	1	0	0	0	7	4
WIND	0	0	10	7	5	0	0	156

Table 2.5: Confusion matrix for Neural Network using Bag of Words

As we can see, the majority of events get classified to their right classes, however, let us take a look at the examples that confuse the model.

True prediction: LIGHTNING Predicted prediction: PRECIP Reference: ['Orages', 'grêle', 'localiser', 'Yonne', 'important', 'dégât', 'culture', 'autour', 'pont-sur-yonne', '200', 'client', 'priver', 'électricité', 'Puisaye', 'YONNE', 'REPUBLICAINE', '29', 'JUN', '2021', 'RAD']

True prediction: WIND Predicted prediction: PRECIP Reference: ['Orages', 'Saint-Etienne', 'Gier', '2618', 'impact', 'foudre', 'relever', 'progrès', '22', 'JUL', '2020']

True prediction: WIND Predicted prediction: HAIL Reference: ['Orages', 'grêle', 'faire', 'casse', 'auvergne', 'Rhône', 'Alpes', 'France', 'Info', '3', 'Jul', '2019']

True prediction: WIND Predicted prediction: LIGHTNING Reference: ['Orages', 'savoie', 'Haute-Savoie', 'foudre', 'frappe', 'transformateur', 'route', 'couper', 'France', 'Bleu', '24', 'OCT', '2022']

True prediction: PRECIP Predicted prediction: LIGHTNING Reference: ['fort', 'intempérier', 'vallée', 'Rhône', 'maison', 'effondrer', 'ardèche', 'dauphiné', '09', 'nov', '2022']

As we can see, the majority of the confusion still comes from the very common word "Orage", thus the model gets confused with similar references in each model. We will see whether this behavior improves with different feature representations.

2.4.2 TF-IDF vector

Now, we will try the second feature representation technique and we will analyze how different are the results attained. With the logistic regression, we get an accuracy of 83.3% and an F1 score of 82.3%, which is very similar to what we attained with the bag of words. Now let us explore the confusion matrix of this logistic regression.

	A	D	H	L	P	S	T	W
AVALANCHE	12	0	0	0	1	0	0	1
DEVIL	0	0	0	0	0	0	0	2
HAIL	0	0	112	0	9	0	0	2
LIGHTNING	0	0	3	78	2	0	0	7
PRECIP	0	0	10	3	26	0	0	13
SNOW	0	0	0	1	0	0	0	0
TORNADO	0	0	1	0	1	0	4	6
WIND	0	0	8	7	2	0	0	161

Table 2.6: Confusion matrix of logistic regression using TF-IDF

As we can see, the results slightly vary, however, the wrongly classified classes remain the same. Now, let us take a look at the examples of misclassified references:

Prediction: HAIL True value: LIGHTNING Reference: ['Orages', 'grêle', 'localiser', 'Yonne', 'important', 'dégât', 'culture', 'autour', 'pont-sur-yonne', '200', 'client', 'priver', 'électricité', 'Puisaye', 'YONNE', 'REPUBLICAINE', '29', 'JUN', '2021', 'RAD']

Prediction: WIND True value: DEVIL Reference: ['saint-sernin', 'BOIS', 'mini-tornade', 'emmèn', 'toiture', 'Creusot', 'Infos', '12', 'april', '2020', 'colère', 'Zeus', 'Facebook', '14', 'april', '2020']

Prediction: WIND True value: PRECIP Reference: ['orage', 'charente', 'rafale', '111.9', 'kilomètre', 'heure', 'semaine', 'pluie', 'minute', 'charente', 'libre', '21', 'june', '2022']

As we can see, we again have some words in misclassified references that could indicate multiple events and thus causing the model to predict an incorrect value.

Now, let us look at the most important features of each class using the same technique as for Bag of Words.

	AVALANCHE	DEVIL	HAIL	LIGHTNING	PRECIP	SNOW	TORNADO	WIND
1	avalanche	tourbillon	grêle	foudre	inondation	neige	tornade	ws
2	skieur	argelès	facebook	incendie	orages	15	oct	meteofrance
3	avalanch	jardinier	jun	maison	inonder	derniere	keraunos	vent
4	savoie	ferté	eyewitness	kachelmannwetter	boue	alsace	français	tempête
5	jan	parasol	2022	feu	pluie	000	observatoire	arbre
6	hautes	bernard	grêlon	foudroyer	eau	priver	2020	com
7	alpes	lacanau	agriculteur	sep	orage	foyer	ef0	mini
8	mort	plage	centre	abattre	pompier	jan	dec	report
9	emporter	poussiérer	météo	2020	intervention	électricité	facebook	feb
10	dec	surprendre	report	aug	09	souffert	nov	coup

Table 2.7: Most important features for Logistic Regression using TF-IDF

They are again very similar to the ones we obtained using Bag of Words. However, they also quite accurately predict the keywords for each of the classes.

Moving on to the Random Forest, we again tune the hyperparameters to obtain the best accuracy. With the parameters set to

1. 'max depth': None,
2. 'min samples leaf': 1,
3. 'min samples split': 2,

4. 'n estimators': 200

we get an accuracy of 83.7% and an F1 score of 83.0%, which again is very close to what we obtained using the Bag of Words. We check out the confusion matrix:

	AVALANCHE	DEVIL	HAIL	LIGHTNING	PRECIP	SNOW	TORNADO	WIND
AVALANCHE	14	0	0	0	0	0	0	0
DEVIL	0	0	0	0	0	0	0	2
HAIL	0	0	111	0	8	0	2	2
LIGHTNING	0	0	3	79	4	0	0	4
PRECIP	0	0	11	5	24	0	0	12
SNOW	0	0	0	0	0	0	0	1
TORNADO	0	0	1	0	1	0	6	4
WIND	0	0	8	5	3	0	0	162

Table 2.8: Confusion matrix for Random Forest using TF-IDF

And as in all of our previous models, mostly overpredicted classes are hail and wind (which are also the most frequent ones as seen in the data analysis part). Let us look at concrete examples:

Actual: LIGHTNING Predicted: WIND Reference: ['pluie', 'inondatier', 'Yonne', '500', 'foyer', 'priver', 'électricité', 'Avallonnais', 'lundi', 'soir', 'YONNE', 'RÉPUBLICAINE', '11', 'may', '2020']

Actual: TORNADO Predicted: WIND Reference: ['photo', 'trombe', 'marine', 'observer', 'entrer', 'Pornic', 'Noirmoutier', 'courrier', 'pays', 'Retz', '29', 'june', '2021']

Actual: LIGHTNING Predicted: WIND Reference: ['arbre', 'foudroyer', 'Aslonnes', 'nuit', 'vendredi', 'samedi', 'république', '10', 'may', '2020']

Actual: PRECIP Predicted: WIND Reference: ['orage', 'charente', 'rafale', '111.9', 'kilomètre', 'heure', 'semaine', 'pluie', 'minute', 'charente', 'libre', '21', 'june', '2022']

Actual: WIND Predicted: HAIL Reference: ['PHOTOS', 'Grêle', 'plui', 'vent', 'isère', 'savoie', 'haute-savoie', 'bien', 'secouer', 'orage', 'dimanche', 'FRANCE', '3', '21', 'JUN', '2021', 'RAD']

Actual: PRECIP Predicted: HAIL Reference: ['Toulouse', 'frapper', 'violent', 'orage', 'grêle', 'mercredi', 'soir', 'Figaro', '20', 'Jun', '2019', 'other', 'source', 'see', 'links']

When there is a word that could indicate different events, a model chooses a more frequent one and thus tends to slightly overfit.

Finally, let us look at the most important features:

Rank	Feature
1	foudre
2	grêle
3	report
4	météo
5	com
6	2022
7	france
8	meteofrance
9	facebook
10	jun

Table 2.9: Most important features for Random Forest using TF-IDF

After careful analysis, we can see that they are almost the same as the ones for Bag of Words. Indeed, so far the performance of both of these representations has been quite similar.

Let us now explore the neural network. After training our network, we get an accuracy of 81.6% - slightly worse than for the bag of words.

The confusion matrix looks like this:

	Predicted Class							
Actual Class	AVALANCHE	DEVIL	HAIL	LIGHTNING	PRECIP	SNOW	TORNADO	WIND
AVALANCHE	14	0	0	0	0	0	0	0
DEVIL	0	0	0	0	1	0	0	1
HAIL	0	0	107	2	11	0	2	1
LIGHTNING	0	0	3	75	6	0	0	6
PRECIP	0	0	10	2	28	0	0	12
SNOW	0	0	1	0	0	0	0	0
TORNADO	0	0	1	0	0	0	6	5
WIND	0	0	11	8	6	0	2	151

Table 2.10: Confusion matrix of neural network using TF-IDF

where we see the same tendencies of overpredicting hail and wind events. Our mispredicted examples look like this:

True prediction: LIGHTNING Predicted prediction: PRECIP Reference: ['Orages', 'grêle', 'localiser', 'Yonne', 'important', 'dégât', 'culture', 'autour', 'pont-sur-yonne', '200', 'client', 'priver', 'électricité', 'Puisaye', 'YONNE', 'REPUBLICAINE', '29', 'JUN', '2021', 'RAD']

True prediction: LIGHTNING Predicted prediction: WIND Reference: ['toiture', 'arracher', 'foudre', 'Douai', 'victime', 'FRANCE', 'bleu', '20', 'JUN', '2021', 'RAD']

True prediction: WIND Predicted prediction: PRECIP Reference: ['Orages', 'Saint-Etienne', 'Gier', '2618', 'impact', 'foudre', 'relever', 'progrès', '22', 'JUL', '2020']

True prediction: WIND Predicted prediction: HAIL Reference: ['Orages', 'grêle', 'faire', 'casse', 'auvergne', 'Rhône', 'Alpes', 'France', 'Info', '3', 'Jul', '2019']

True prediction: DEVIL Predicted prediction: WIND Reference: ['saint-sernin', 'BOIS', 'mini-tornade', 'emmèn', 'toiture', 'Creusot', 'Infos', '12', 'april', '2020', 'colère', 'Zeus', 'Facebook', '14', 'april', '2020']

where again we find the same common words and mispredictions for smaller classes, except for some, where we have words that can be tricky to identify ('foudre' would be an example).

By switching our feature representation to TF-IDF we did not achieve better accuracy or any new revelations. Therefore, we move on to a more intricate feature representation technique, that will capture the semantic meaning of a sentence hopefully letting us achieve greater accuracy.

2.4.3 SBERT transformer

In order to capture the semantic meaning within our references, we employ the SBERT embedding technique. SBERT stands for Sentence-BERT and it is a modification of the pre-trained BERT network that allows to obtain sentence embeddings. Using this technique, we expect to achieve better accuracy than with the Bag of Words or TF-IDF, since it should understand better the meaning and context of the full sentences.

Our first advantage is that we do not have to separate the dataset by languages as SBERT has models for multilingual embeddings.

Thus, after creating the embeddings we run the logistic regression and we get an accuracy of 80.6%, which is slightly lower than with other representations.

Let us look at the confusion matrix.

	A	D	H	L	P	S	T	W
AVALANCHE	12	0	0	0	0	0	0	1
DEVIL	0	0	0	1	1	0	0	1
HAIL	0	0	176	1	4	0	0	17
LIGHTNING	0	0	2	76	2	0	1	4
PRECIP	0	0	12	3	27	0	0	23
SNOW	0	0	1	0	0	0	0	0
TORNADO	0	0	3	0	2	0	7	10
WIND	0	0	9	5	14	0	0	188

Table 2.11: Confusion matrix for SBERT for Logistic Regression

As always, we see the same tendencies of guessing the most frequent features and unfortunately, SBERT embeddings did not give us any more accuracy.

Let us look at the misclassified examples: Actual: LIGHTNING Predicted: WIND Reference: "Les très belles photos des chasseurs d'orages de Météo Centre", FRANCE BLEU, AUG

Actual: TORNADO Predicted: WIND Reference: Tornade à Champsac ()] - //, Ouest Orages, n.d.. Présage des Vents (on Facebook), Jan : Tornade de Champsac () - janvier . Photos by A. Rivet.

Actual: WIND Predicted: HAIL Reference: Eyewitnesses reports in comments via Météo Sorguaise et Avignonnaise (on Facebook), AUG .

Actual: HAIL Predicted: PRECIP Reference: Violents orages, pluies diluviennes et inondations du au septembre, Meteo Paris, Sept . RAD. Eyewitness report via Météo42 (on Facebook), SEP .

As we have the references printed, we can truly see the noise and uncertainties. It could be that noise is one of the reasons why SBERT does not perform better.

Let us now tune the hyperparameters of the Random Forest. In this case, we get

1. 'max depth': 20,
2. 'min samples leaf': 2,
3. 'min samples split': 2,

4. 'n estimators': 200

and an accuracy of 78.1% and an F1 score of 76.1%, which again is slightly worse than our previous results. Our confusion matrix looks like this:

	AVALANCHE	DEVIL	HAIL	LIGHTNING	PRECIP	SNOW	TORNADO	WIND
AVALANCHE	11	0	0	0	0	0	0	2
DEVIL	0	0	1	0	0	0	0	2
HAIL	0	0	174	3	5	0	0	16
LIGHTNING	0	0	3	73	3	0	0	6
PRECIP	0	0	17	4	14	0	0	30
SNOW	0	0	0	0	0	0	0	1
TORNADO	0	0	3	1	0	0	4	14
WIND	0	0	18	7	6	0	0	185

Table 2.12: Confusion matrix for SBERT for Random Forest

and some of the misclassified examples: Actual: HAIL Predicted: PRECIP Reference: Orages de grêle très localisés dans lYonne dimportants dégâts dans les cultures autour de PontsurYonne et clients privés délectricité en Puisaye LYONNE RÉPUBLICAINE JUN RAD

Actual: DEVIL Predicted: WIND Reference: Lacanau quel est ce tourbillon qui a fait senvoler tous les parasols de la plage Sud Ouest Jul

Actual: WIND Predicted: HAIL Reference: Intempéries Record une rafale de vent à kmh enregistrée à Montpellier Actufr June

Actual: WIND Predicted: PRECIP Reference: Un supermarché frappé par la foudre et inondé à Thyez une cinquantaine dinterventions pour les pompiers LE DAUPHINÉ AUG

Actual: TORNADO Predicted: WIND Reference: StevenTual off twitter NOV Impressionnant une trombe marine observée à Penmarch Vidéos Le Telegramme NOV Une trombe marine se forme au large du Finistère en ce jour de Toussaint OuestFrance NOV Trombe marine sur la côte du Finistère le 1er novembre Observatoire Keraunos NOV Eyewitness report via Penmarch on Facebook NOV

where we see the same tendencies as before and even some of the same references that were misclassified. Thus, we can see that some of these simply do not fit our project.

Finally, let us look into the neural networks. After running it on the SBERT embeddings, we get an accuracy of 80.6%, which again is not better than for the previous models.

We check again the confusion matrix, where we see the same tendencies as before.

True prediction: HAIL Predicted prediction: PRECIP Reference: Var létat de catastrophe naturelle reconnu à Hyères et au Pradet après les orages du août BFM TV AUG

True prediction: LIGHTNING Predicted prediction: PRECIP Reference: Orages et inondations en Normandie des centaines dinterventions des pompiers ACTUfr JUN

True prediction: WIND Predicted prediction: HAIL Reference: Météo Nevers v Facebook SEP

	Predicted Class							
Actual Class	AVALANCHE	DEVIL	HAIL	LIGHTNING	PRECIP	SNOW	TORNADO	WIND
AVALANCHE	12	0	1	0	0	0	0	0
DEVIL	0	0	0	1	1	0	0	1
HAIL	0	0	187	0	4	0	1	6
LIGHTNING	0	0	3	73	5	0	1	3
PRECIP	0	0	17	1	29	0	2	16
SNOW	0	0	1	0	0	0	0	0
TORNADO	0	0	3	0	2	0	10	7
WIND	0	0	16	6	13	0	6	175

Table 2.13: Confusion matrix of neural network using SBERT

True prediction: PRECIP Predicted prediction: WIND Reference: Chasseneuil la RN rouverte à la circulation après une inondation encore foyers sans électricité La Charente Libre June

where we see similar words that get misclassified.

Therefore, as we have seen through the process, there are multiple keywords that help our models identify which severe weather event the reference is talking about. However, this also leads us to another pressing question - how do we make that our models do not confuse references about not-weather events or about weather events that are not severe if they contain these keywords? This is what the second part of the project helped us explore.

Chapter 3

Binary classifier for false data

Besides classifying the event types to references, it is crucial to have a classifier that could distinguish between severe weather events and not or even other articles that contain keywords.

Thus, for the second part of the experiment, the goal was to find a good database of false data that would be good false data in order to build a binary classifier. We only keep the French part of our old dataset and retrieve almost 500 false data references in French which will help us complete the task.

3.1 Techniques Used

As we have already explored many techniques in the previous part, we will only shortly brush up on them in this chapter focusing more on the results attained.

We take a random subset of the French data of 500 entries. We then combine the 500 false French references with 500 true French references from the initial dataset to create a balanced dataset of 1000 French references for binary classification. For preprocessing, we tokenize the text data, remove punctuation and stopwords, and lemmatize the remaining tokens. We represent each reference using Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) vectorization.

We train Logistic Regression (LR) classifier on the vectorized representations. LR is a simple model providing a probabilistic prediction and before in the experiment, we concluded it does work well even when compared to other, more intricate models.

To evaluate LR, we perform 10-fold cross-validation. We randomly split the 500 true French references into 10 equal folds, trained the model on 9 folds (90 percent of the data) and test on the held-out fold. We repeat this process 10 times, using each fold as the test set once. Cross-validation limits overfitting and provides a robust evaluation of the model's performance [25].

3.2 Process and Results

As we have seen, for our data we get satisfactory results using BoW and TF-IDF vectors with Logistic Regression, thus here we decided to limit ourselves to simpler techniques, as they bring us the same accuracy as more intricate ones.

Let us explore the references from real and false datasets: *Orages mois plui 1 heure Civray 100 intervention pompier nouvelle république 17 JUN 2021 RAD*

dépression Nino 3 m neige pluie torrentiel sable

As can be seen, they are quite similar and use keywords found we found for the first part of the experiment. After converting our references using Bag of Words and feeding it to the Logistic Regression, using the 10-fold technique, we retrieve an accuracy of 99.4%. As we can see, binary classification is truly an easy task.

Our incorrectly classified data are these two samples:

Predicted: 1 True: 0 Reference: météo 2022 lodévoi ancrée Predicted: 1 True: 0 Reference: 2022 bien être l ' année chaude France

As we can see, they both contain keywords that we identified in the previous part of the project - "2022", "France", "météo". Thus, the model does not work perfectly, but we can also notice that the references indeed contain many same words.

When we move on to the TF-IDF vector, we get a 98.9% accuracy, which again is a great performance from the model.

Some examples of our misclassifications:

1. Haute-Savoie glissement terrain bloquer route Thyez dauphiner Libéré 15 July 2021 for suscriber only (True label: 1, Predicted label: 0)

2. grêle sud-est orage l ' Ouest (True label: 0, Predicted label: 1)

3. météo France surprendre (True label: 0, Predicted label: 1)

4. Mickael B. observatoir ciel Orageux Tornade Médoc 2018 2019 (True label: 1, Predicted label: 0)

5. match xv never reporter cause vent Péméja never furieux réaction vidéo charente libre 01 March 2020 (True label: 1, Predicted label: 0)

6. témoignage voisin n ' fumée c ' toit partir voir pire REDON MAVILLE 24 nov 2022 (True label: 1, Predicted label: 0)

Again, we can see that most are very short and thus hard to classify. Also, most are classified as severe even though are not contain multiple keywords. However, despite some setbacks, our model works very well at least on our dataset.

In order to test it better, we should acquire a larger and same-sourced database to make the research more precise.

Chapter 4

Conclusion & Outlook

In this project, we explored the classification of severe weather events using textual data from the European Severe Weather Database. Several machine learning models were tested, including SBERT, fully-connected neural networks, logistic regression, and support vector machines. Overall, all models tested achieved good performance with at least 80% accuracy, indicating the textual data contains meaningful signals to distinguish between different types of severe weather events, however, none of the models or feature representations especially stood out. We have also seen that the data collected by volunteers is especially noisy and is not held to any standard, thus making any sort of processing quite a difficult task.

We also developed a binary classifier to filter out false data that contains key weather-related terms but does not actually describe a severe weather event. This classifier achieved over 95% accuracy, making it suitable and a starting point for automatization of Severe Weather Event Database automatization and perhaps further expansion. In order to better its performance, we should take references from identical sources, so that such things as date and year present could not indicate whether the reference is false data.

These two experiments can definitely show that if a proper cleaning of data was done and an efficient way was found to extract key information from the articles, there are models that could take over the volunteer work.

The results demonstrate the feasibility of automatically classifying severe weather events based on witness reports and social media data. With further refinement, such a system could help filter and categorize large amounts of unstructured data to aid forecasters and researchers. The machine learning models could also be expanded to classify sub-categories of events or detect the specific weather phenomena mentioned in a report.

There are several promising avenues for future work based on this project. With a larger dataset, deep learning models may achieve even higher accuracy in classifying severe weather events and sub-categories. Contextualized word embedding models like BERT

[22] and RoBERTa [26] should also be explored as they have achieved state-of-the-art results on various text classification tasks.

The binary false data classifier could be expanded to also classify reports that do not contain any mention of severe weather and filter them out. Event detection algorithms could also be developed to automatically extract reports of severe weather events from Internet data sources like social media streams. Also, more data could be extracted about the event, such as date and time and other parameters. This could definitely benefit climate research and its impact on our planet.

By improving the accuracy and capabilities, automated techniques like these could help build up valuable resources to study severe weather events and improve forecasting accuracy and warning systems. Overall, natural language processing and machine learning provide a wealth of opportunities for processing disaster-related data that remains largely untapped. With enhanced algorithms and the exponential growth of available data, these techniques are poised to become even more valuable tools for researchers and practitioners working to study and mitigate severe weather events.

References

[1] Pieter Groenemeijer, M Kuehne, Z Liang, and N Dotzek. New capabilities of the european severe weather database. *In 5th European conference on severe storms, Landshut, Germany*, pages 311–312, 2009.

[2] Vittorio A Gensini and Thomas L Mote. Downscaled estimates of late 21st century severe weather from ccsm3. *Climatic Change*, 129(1):307–321, 2015.

[3] Davide Buscaldi and Irazu Hernandez-Farias. Sentiment analysis on microblogs for natural disasters management: a study on the 2014 genoa floodings. *Proceedings of the 24th international conference on world wide web*, pages 1185–1188, May 2015.

[4] Elena Simperl Sophie Parsons, Peter M Atkinson and Mark Weal. Thematically analysing social network content during disasters through the lens of the disaster management lifecycle. *Proceedings of the 24th international conference on world wide web*, pages 1221–1226, 2015.

[5] H. E. Brooks C. A. Doswell and M. P. Kay. Climatological estimates of daily local nontornadic severe thunderstorm probability for the united states. *Weather and Forecasting*, 20(4):577–595, 2005.

[6] Ilan Kelman Jessica Mercer and Lorin Taranis. Framework for integrating indigenous and scientific knowledge for disaster risk reduction. *Climate Change Modeling, Mitigation, and Adaptation*, 34(1):214–239, 2009.

[7] Matthew Honnibal, Ines Montani, Sam Van Landeghem, and Adriane Boyd. SpaCy: Industrial-strength natural language processing in Python, 2020.

[8] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied Logistic Regression*. Wiley, 2013.

[9] J. S. Cramer. The origins of logistic regression. Technical report, Tinbergen Institute, 2002.

[10] Jesse Read. *Introduction to Machine Learning, CSE204 Course Notes*. 2023.

[11] Alan Agresti. *Categorical Data Analysis*. Wiley, 2002.

[12] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer Science Business Media, 2006.

[13] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[14] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[15] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[16] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing Management*, 24(5):513–523, 1988.

[17] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

[18] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[19] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[20] Gerard Salton and Michael J McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[21] Christopher D Manning, Prabhakar Raghavan, and Hinrich Sch"utze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[23] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[24] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzm'an, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.

[25] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. In *Encyclopedia of database systems*, pages 532–538. Springer, 2009.

[26] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.