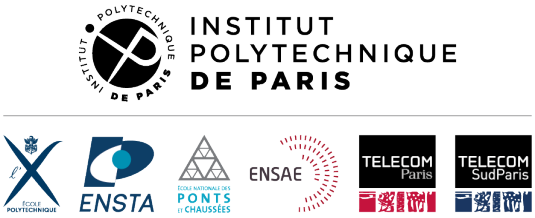
Misinformation Detection : from basics NLP techniques to deep learning methods
Paper : "*Exploring the generalizability of Fake News Detection Models*" by Hoy and Koulouri, 2023.

Amine Razig
ENSAE - Polytechnique
amine.razig@polytechnique.edu

Mohamed Keteb
ENSAE Paris - Université Paris-Saclay
mohamed.keteb@ensae.fr



Project summary

Misinformation detection is a critical problem in the field of Natural Language Processing (NLP). This report explores various methods for detecting fake news using classical machine learning and modern deep learning techniques. The project involves data pre-processing, feature extraction, and model evaluation. The results demonstrate the effectiveness of different approaches in identifying misinformation. We first demonstrate that classical machine learning models achieve high accuracy across various feature extraction techniques when evaluated on a holdout set—an unseen portion of the same dataset used for training. While these results may initially seem promising, they are not entirely unexpected. The core issue this project aims to highlight is the problem of generalization. When tested on data from different distributions or sources, the models exhibit a significant drop in accuracy, revealing their limited ability to generalize beyond the training data. This project is inspired by the study presented in the article of Hoy and Koulouri [2023], which investigates the generalization of fake news detection models through cross-dataset evaluation. The work explores the impact of different feature extraction methods, machine learning models, and datasets. All our code and experimental results are available in this Github repository[1], ensuring the reproducibility of our work.

[1] **GitHub Repository link** : $https://github.com/arazig/Misinformation-detection$

1 Introduction

Fake news has become increasingly prevalent with the rise of data-driven infrastructures and the widespread use of digital platforms. Misinformation now impacts a wide range of fields, including politics, health, and science, with a notable surge during global events such as the COVID-19 pandemic. Additionally, the emergence of Large Language Models (LLMs) has introduced new challenges, as these models can be exploited to generate convincing yet false content at scale. In this context, machine learning models appear to offer a promising solution for the detection and classification of fake news. Their capacity to learn patterns from large textual datasets makes them suitable candidates for this task. However, their application is not without limitations. In this project, we aim to explore both the strengths and the shortcomings of various machine learning approaches in the context of fake news classification.

Our primary objective is to demonstrate that classical machine learning models—specifically Logistic Regression, Support Vector Machines (SVM), Naive Bayes, and Random Forest—perform remarkably well in the task of fake news classification. As shown in the article of Hoy and Koulouri [2023], these models can achieve accuracies of over 90% when trained and evaluated on the same dataset, using various feature extraction techniques such as Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), and Word2Vec, when applicable to the model architecture.

Following this, we extend our analysis to more advanced deep learning approaches, including Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs). We then investigate the generalization capabilities of these models by evaluating them on a richer and more diverse dataset, which differs significantly from the training data—primarily composed of politically oriented fake news.

Finally, we assess the performance and generalization ability of a Large Language Model (LLM), comparing it to both the traditional and deep learning models discussed earlier. This comparison allows us to critically examine how different modeling paradigms handle the challenge of misinformation detection across domains.

2 Related work

The systematic literature review (SLR) conducted by the authors analyzed 1063 publications on fake news detection from 2016 to 2020 (with an ongoing extension to 2021 and 2022). The SLR revealed that most fake news detection methods rely on content-based textual features and typically achieve promising results, with an average accuracy of around 80%. However, the categorization of textual features remains inconsistent across studies. For this reason, the review broadly classifies them into two main types.

Word-level representations are techniques that transform textual input into numerical vectors and include Bag-of-Words, TF-IDF, and word embeddings. ***Linguistic cues*** features are derived from textual analysis, including syntactic and stylistic elements such as sentence complexity, Part of speech (POS) tags, average word length, named entities, sentiment scores, and pronoun or verb frequency. Note that in our study, we will focus exclusively on word-level representations, which are detailed later.

Several studies have investigated the performance of classical machine learning models using these features. For example, Kaur et al. [2020] evaluated Bag-of-Words and TF-IDF with classifiers like Naïve Bayes, Logistic Regression, SVMs, Gradient Boosting, and Neural Networks, reporting accuracies up to 98.3% on a Kaggle dataset. Similarly, Poddar et al. [2019] used TF-IDF and SVMs, achieving 92.8% accuracy. These results highlight the strong potential of word-level features when combined with effective classification algorithms.

Gautam and Jerripothula [2020] examined the generalization capability of fake news detection models across two distinct news domains: celebrity and political news. Their results revealed a significant performance drop when models were trained and tested on different domains. Specifically, a model trained on political news and tested on celebrity news experienced a 39% drop in accuracy, while the reverse scenario resulted in an 8% drop. Although the celebrity-trained model appeared more robust, its baseline accuracy on the hold-out set was only 78%, compared to 95% for the political model, limiting the extent of the observed drop.

Similarly, Blackledge and Atapour-Abarghouei [2021] investigated cross-topic generalization using the ISOT by Ahmed et al. [2017] and Ahmed et al. [2018] and Combined Corpus datasets (CC) by Khan et al. [2021]. While ISOT primarily contains political news, CC spans various domains including healthcare, sports, and entertainment. These datasets are also significantly larger (44,898 and 79,548 samples, respectively) than those used by Gautam and Jerripothula [2020], making them more suitable for evaluating generalizability. Their study found that models trained and tested on the same dataset achieved over 90% accuracy. However, when evaluated across datasets, accuracy dropped by about 25% when transferring from ISOT to CC, and by around 15% in the opposite direction.

In our study, we adopt a similar approach to assess cross-topic generalizability by training and testing models using both ISOT and CC datasets

3 Methodology

3.1 Data Preprocessing

The dataset consists of fake and true news articles. The preprocessing steps include:

- **Removing special characters, numbers, and stopwords**: Special characters (e.g., @, , $, %, etc.) and numbers do not typically carry meaningful information for text classification tasks. They might create noise in the model, reducing its ability to focus on relevant content. Stopwords (common words like "the," "and," "is," "in," etc.) are frequently used words that generally do not contribute to the understanding of the content in the context of text classification. Removing stopwords helps reduce computational load and allows the model to focus on more informative words.
- **Converting text to lowercase**: Text data can come in different letter cases (e.g., "Fake" and "fake" may be treated as different words). Converting all text to lowercase ensures consistency and avoids treating the same word differently based on its case, improving the model's accuracy by focusing solely on the word itself.
- **Applying lemmatization to reduce words to their base forms**: Lemmatization is the process of reducing words to their base or root form (e.g., "running" becomes "run" and "better" becomes "good"). This step helps ensure that different forms of the same word are treated as the same feature, reducing the dimensionality of the data and improving the model's ability to generalize across different variations of the same concept.

3.2 Feature Extraction

In Natural Language Processing (NLP), feature extractors are essential tools that transform textual data into numerical representations that can be processed by machine learning models. Since models operate on numerical input, extractors serve to create machine-readable formats.

3.2.1 Bag of Words (BoW)

The Bag of Words model is one of the simplest and most intuitive extraction methods. It converts a collection of text documents into fixed-length vectors based on word occurrence. In BoW, each unique word in the entire corpus is assigned a specific position in a vector, and each document is then represented by the frequency (or presence/absence) of these words. Note that this method ignores grammar and word order.

3.2.2 Term Frequency-Inverse Document Frequency (TF-IDF)

On the other hand, TF-IDF refines the BoW approach by weighting the frequency of a word in a document (Term Frequency) against its frequency across all documents (Inverse Document Frequency). This helps downweight common words that carry less discriminative power—such as "and", "the", and "for"—and upweight more informative, rare words.

- t is the word,
- d is a document,

- D is the collection (corpus) of documents,
- $f_{t,d}$ is the raw count of term t in document d,
- N is the total number of documents in the corpus,
- $|\{d \in D : t \in d\}|$ is the number of documents in which the term t appears.

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D)$$

$$\text{tf}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

$$\text{idf}(t, D) = \log\left(\frac{N}{|\{d \in D : t \in d\}|}\right)$$

The TF-IDF score of a word increases with the number of times it appears in a document but decreases with its frequency across the corpus. This makes TF-IDF particularly effective for our fake news classification problem where distinguishing key terms is important.

3.2.3 Word2Vec

Unlike TF-IDF and Bag of Words (BoW), which represent words based solely on their frequency and do not account for the order or context in which words appear, Word2Vec is designed to capture semantic relationships between words by considering their context in a given corpus.

Word2Vec is a more advanced feature extraction that generates dense vector representations for words, where words with similar meanings are represented by vectors that are closer in the vector space. This is achieved through training on large text corpora using the Continuous Bag of Words (CBOW) and Skip-gram model represented in Figure 1. Both models rely on a neural network architecture, and through training, they adjust the weights of the network to produce vector representations that minimize prediction errors.

- In the CBOW model, the objective is to predict a target word based on its surrounding context words. For example, in the sentence "The cat sits on the mat," CBOW tries to predict "sits" given the context words "The", "cat", "on", "the", and "mat".
- In contrast, the Skip-gram model works by using a target word to predict its surrounding context. Using the same example, Skip-gram would take "sits" as the input and predict "The", "cat", "on", "the", and "mat" as the context words.

Note that each context word is represented by a one-hot encoded vector, where the vector's length is equal to the size of the vocabulary. Once the vector representations for each word are obtained, it becomes possible to compute an embedding for an entire text. Several common strategies are used for this purpose:

1. *Averaging Word Embeddings* : One of the most straightforward methods involves averaging the embeddings of all words in the text. Each word contributes equally to the final representation.

$$\text{TextEmbedding} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{w}_i$$

   Where $\mathbf{w}_i$ is the embedding of the i-th word, and n is the total number of words in the text. This method is simple to implement and computationally efficient, which is why we chose to use it in our project. However, it assumes that all words are equally important, which can lead to a loss of semantic nuance and disregards the order of words in the text.

2. *TF-IDF Weighted Averaging* : This aggregation involves weighting each word's embedding by its importance in the text, typically using the Term Frequency–Inverse Document Frequency (TF-IDF) metric. This allows more informative words to contribute more to the final text representation.

$$\text{TextEmbedding} = \frac{\sum_{i=1}^{n} \text{TF-IDF}(w_i) \cdot \mathbf{w}_i}{\sum_{i=1}^{n} \text{TF-IDF}(w_i)}$$

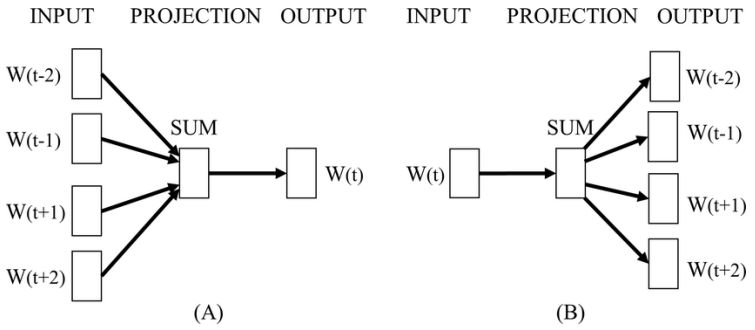


Figure 1: CBOW (A) and Skip-gram (B)

Where TF-IDF(w_i) is the TF-IDF weight for the word w_i, and $\mathbf{w}_i$ is its corresponding embedding. Similarly, TF-IDF can be seen as an enhancement of the Bag-of-Words model, and when used for averaging word embeddings, it improves upon simple uniform averaging by assigning greater weight to more informative words. Nevertheless, like basic averaging, it fails to capture the sequential structure and contextual nuances of language.

The key takeaway is that Word2Vec provides more sophisticated representations by incorporating the context in which words appear. However, as noted in Hoy and Koulouri [2023], it does not account for word order and generates the same embedding for words that are morphologically identical but semantically different. For example, the word "left" may refer to the past tense of "to leave" or the directional term "left." This illustrates that Word2Vec is context-independent, as it does not distinguish between such differing usages.

3.2.4 Sentence embedding (BERT)

Finally, the last feature extractor we propose to use in this project is a contextual embedding : Bert. It provide word representations that depend on the specific usage of words within a sentence. Unlike traditional static embeddings like Word2Vec or Glove which assign the same vector to a word in all contexts, BERT adjusts the embedding of a word depending on its surrounding words.

This should makes BERT particularly effective for handling polysemous words those with multiple meanings which are common in misleading or ambiguous content. For instance, the word *"charge"* can refer to a legal accusation or to battery power. BERT can disambiguate such meanings by relying on context.

The key to BERT's contextual power lies in its bidirectional transformer architecture (cf. Appendix 5). This means it analyzes both the left and right context of each word when generating embeddings. As a result:

- Word order and sentence structure are taken into account.
- The same word will have different embeddings in different contexts.
- The model captures sentence-level meaning rather than just isolated words.

When using BERT, each transformer layer outputs contextual embeddings for each token. Two common strategies for selecting an embedding are:

- Using the final layer embedding.
- Averaging the last four layers to get a more stable, semantically rich representation.

For sentence-level classification tasks like misinformation detection, BERT introduces a special token `[CLS]` at the beginning of the input. The final hidden state of this token serves as a representation of the whole sentence. This vector is passed to a classification head (often a simple linear layer) that outputs the prediction, such as *Fake* or *Real*.

In practice, the BERT encoder is often used as a frozen feature extractor. This means that the embedding layers are not updated during training. Instead, only the final classification layer is trained on the task-specific dataset, allowing for efficient fine-tuning.

3.3 Modeling

In this project, we investigate the generalization capabilities of classical machine learning models for classification, as well as more sophisticated deep learning approaches.

3.3.1 Classical classification models

- **Naive Bayes**

The Naive Bayes model is a machine learning algorithm based on Bayes' Theorem. In the context of misinformation detection, Naive Bayes can be used as a text classification tool to determine whether a given article is likely to be misinformation or not.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

In this formula:

- $P(A|B)$ is the probability of event A given B—in this case, the probability that an article is misinformation given the words it contains.
- $P(B|A)$ is the probability of observing the words B in a misinformation article.
- $P(A)$ is the prior probability that an article is misinformation.
- $P(B)$ is the probability of observing the words B in any article.

Under the assumption of word independence (the "naive" assumption), the formula becomes:

$$P(\text{Fake}|m_1, m_2, \ldots, m_n) = \frac{P(m_1|\text{Fake}) \cdot P(m_2|\text{Fake}) \cdot \ldots \cdot P(m_n|\text{Fake}) \cdot P(\text{Fake})}{P(m_1, m_2, \ldots, m_n)}$$

During training, since the data is labeled, the model uses the training set to compute the conditional probabilities of each word given a target class. Then, at prediction time, Bayes' Theorem is applied to calculate the likelihood that a new article belongs to each class (e.g., *Fake* vs. *Real*). For each class, the model multiplies the conditional probabilities of the observed words with the prior probability of the class and divides by the overall probability of the word set. Finally, the article is assigned to the class with the highest resulting probability.

- **Logistic Regression :**

Let us denote by $\mathcal{X} := \mathbb{R}^d$, $\mathcal{Y} := \{-1, 1\}$. If we are given a dataset $(X_i, Y_i)_{i=1}^n \in (\mathcal{X} \times \mathcal{Y})^n \sim P^{\otimes n}$. We define the Bayes risk for the binary loss metric $\forall y, y'\ c(y, y') = \mathbf{1}_{y \neq y'}$ as follows

$$\forall f \in \mathcal{F} := \mathcal{Y}^{\mathcal{X}}, \quad \mathcal{R}(f) = \mathbb{E}[c(f(X), Y)]$$

$f^\star$ is a Bayes predictor if it minimizes the Bayes risk

$$f^\star \in arg\min_{f \in \mathcal{F}} R_P(f)$$

In this specific case, we have $f^\star(x) = 2 \cdot \mathbf{1}_{\zeta(x) > \frac{1}{2}} - 1 = \text{Sign}(2 \cdot \zeta(x) - 1)$ where $\zeta(x) := P(Y = 1 \,|\, X = x)$ and $\text{Sign}(t) := 2 \cdot \mathbf{1}_{t>0} - 1$.

The Logistic model assumption is

$$\exists \boldsymbol{\theta} \in \mathbb{R}^d,\ \forall x \in \mathcal{X}, \quad \mathbb{P}[Y|X = x] = S(Y f_{\boldsymbol{\theta}}(x))$$

with $\boldsymbol{\theta} = (a, b) \in \mathbb{R}^{d+1}$, $f_{\boldsymbol{\theta}}(x) = a + b^\top x$ and $S(t) := \frac{1}{1+e^{-t}}$, $t \in \mathbb{R}$.

We estimate $\boldsymbol{\theta}$ with the maximum Likelihood estimator $\hat{\boldsymbol{\theta}}$. The plug-in estimator of the Bayes predictor defines the Logistic regressor

$$\boxed{\hat{f}(x) = \text{Sign}(2 \cdot S(f_{\hat{\boldsymbol{\theta}}}(x)) - 1) = \text{Sign}(f_{\hat{\boldsymbol{\theta}}}(x))}$$

- **Support Vector Machines (SVM) :**

The central intuition behind Support Vector Machines (SVMs) can be effectively illustrated by the following Figure 2. he idea is that in the original input space, the data is not always linearly separable. Therefore, it is projected into a higher-dimensional space using a transformation function called a feature map, denoted by ϕ. For example, if our input data lies in $\mathbb{R}^d$, such as word embeddings obtained through Word2Vec, then ϕ is a mapping from $\mathbb{R}^d$ to $\mathbb{R}^{d'}$ where $d' \gg d$. By transforming the data into this higher-dimensional space, the hope is that it becomes linearly separable, or at least nearly separable, while allowing for some degree of misclassifications.Now, let's explain the math step by step, in the case where the mapped data $\mathcal{D} = (\phi(X_i), Y_i)_{i=1}^n$ is perfectly separable, meaning that there exists a $w \in \mathbb{R}^{d'}$ such that $Y_i(\langle \phi(X_i), w \rangle + b) \geq 0$ with a bias b to have an affine hyperplane. The margin of w is then

$$marg_{\mathcal{D}}(w) := \min_{i=1,\dots,n} d(\phi(X_i), H_w) \tag{1}$$

where $H_w := \{z \in \mathbb{R}^{d'}, \langle z, w \rangle + b = 0\}$ which is an a affine hyperplane. the Idea is to take the separation that maximizes the margin meaning

$$\max_{w \in \mathbb{R}^{d'}} marg_{\mathcal{D}}(w) \tag{2}$$

one can show that (2) is equivalent to the following quadratic problem

$$\begin{aligned} \min_{w \in \mathbb{R}^d} \quad & \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad & Y_i(\langle w, \phi(X_i) \rangle + b) \geq 1, \quad \forall i = 1, \dots, n \end{aligned}$$

. This problem is called Hadr-SVM, however perfect separability is not always the case so we relax the last problem by allowing some misclassifications as follows

$$\begin{aligned} \min_{w \in \mathbb{R}^{d'},\, b \in \mathbb{R},\, \boldsymbol{\xi} \in \mathbb{R}^n} \quad & \frac{1}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^n \xi_i \\ \text{subject to} \quad & Y_i\left(\langle w, \phi(X_i) \rangle + b\right) \geq 1 - \xi_i, \quad \forall i = 1, \dots, n \\ & \xi_i \geq 0, \quad \forall i = 1, \dots, n \end{aligned}$$

One downside of using a feature map is the difficulty of explicitly computing $\phi(\mathbf{X}_i)$. To address this, we use the dual formulation of the SVM, which involves only inner products of the form $\langle \phi(\mathbf{X}_i), \phi(\mathbf{X}_j) \rangle$. These inner products can be efficiently computed using a kernel function, as every feature map implicitly defines a corresponding kernel, and vice versa. By choosing a specific kernel function, we are effectively fixing the feature map without ever needing to compute it explicitly.

$$\begin{aligned} \max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} \quad & \sum_{i=1}^n \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C \quad \text{for } i = 1, \dots, n \end{aligned}$$

We can find the optimal α with quadratic programming and derive the SVM predictor :

$$\boxed{f_{\text{SVM}}(\mathbf{x}) = \text{Sign}\left(\sum_{i=1}^n \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right)}$$

To choose a kernel, one can either rely on prior knowledge about the data or select the kernel that yields the best generalization performance based on validation accuracy.

- **Random Forest :**

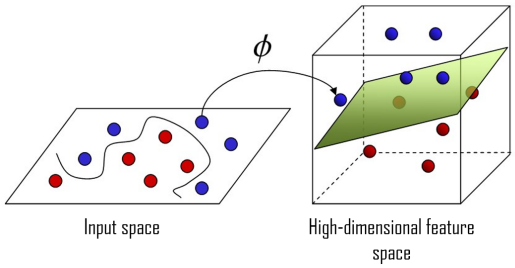


Figure 2: Linear separability in a higher-dimensional space

A decision tree is a model used for classification tasks, like fake news detection, where the data is split at each node based on feature values to create a tree-like structure of decisions. However, a single decision tree can be prone to overfitting, especially with noisy data. Random Forest addresses this issue by using bagging (Bootstrap Aggregating), where multiple decision trees are trained on different subsets of the data (with replacement) and their predictions are aggregated to improve accuracy and reduce variance. The key advantage of Random Forest is its ability to provide more reliable and robust predictions by averaging out errors from individual trees. Its strengths include high accuracy, versatility meaning that it can handle different types of data, and resistance to overfitting, but it can be computationally expensive and less interpretable compared to a single decision tree.

3.3.2 Neural nets methods

- **Long Short-Term Memory (LSTM)**

Long Short-Term Memory (LSTM) networks are a particular class of Recurrent Neural Networks (RNNs), designed specifically to address some of the fundamental limitations of traditional RNNs. Before presenting the LSTM architecture in detail, it is important to first understand the general motivation and mechanics behind RNNs.

Recurrent Neural Networks were introduced to model sequential data, such as time series, text, or speech. Unlike feedforward neural networks, RNNs are equipped with a feedback loop, allowing information to persist across time steps. This makes them well-suited for tasks where the current output depends not only on the current input, but also on previous inputs which is a common scenario in NLP.

RNNs face practical training challenges. The most prominent among them is the vanishing (or exploding) gradient problem during backpropagation. When computing gradients over long sequences, the repeated application of the chain rule can cause gradients to either become very small (vanish) or very large (explode), making the learning impossible.

LSTM is an architecture designed to tackle the problem of vanishing or exploding gradients by maintaining both long-term and short-term dependencies. Long-term and short-term memory are updated through 3 distinct gates: the input gate, forget gate and output gate represented in Figure 3. For example, the **forget gate** takes the current input X_t and the previous short-term memory h_{t-1}, passes them through a sigmoid activation function, which determines the proportion of the previous long-term memory C_{t-1} that should be retained.

Unsurprisingly, the three-gate mechanism makes training LSTMs computationally more expensive compared to standard RNNs. Additionally, they may still face challenges in capturing very long-term dependencies, particularly when compared to more recent architectures such as Transformers, which will be discussed later in this report.

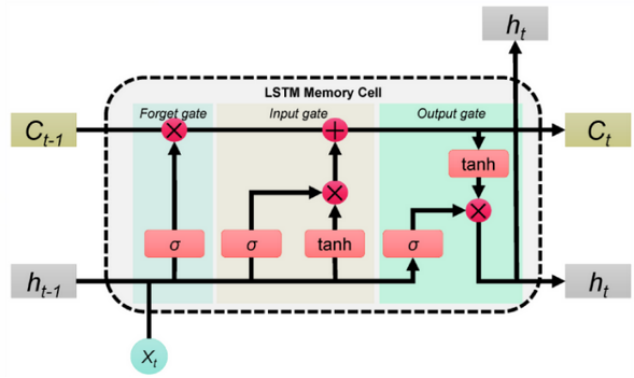


Figure 3: LSTM Architecture

Note that to classify fake news using an LSTM, the input text is converted into word embeddings and processed sequentially through the LSTM. The final hidden state h_T, which captures the overall context of the text, is passed to a dense layer with a sigmoid activation to output a binary prediction.

- **Classification using a Convolutional Neural Network**

Then we decided to use another neural net model. We use a Convolutional Neural Network (CNN) to automatically classify articles. While CNNs were originally designed for image processing tasks, they have also proven to be effective in Natural Language Processing due to their ability to extract local semantic patterns in sequences of text. Intuitively, the text can be simply viewed as an 1D signal.

Vector Representation of Articles As we explained before, each article is represented as a sequence of words $(w_1, w_2, \ldots, w_L)$, where L is the fixed maximum sequence length. And each word w_i need to be projected into a dense vector space of dimension d, here we simply use an `Embedding` layer.

We denote by $\mathbf{m}_i \in \mathbb{R}^d$ the embedding vector corresponding to the word w_i. The entire article is thus represented as a matrix $\mathbf{A} \in \mathbb{R}^{L \times d}$, defined as:

$$\mathbf{A} = \begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_L \end{bmatrix} = \begin{bmatrix} \text{Embedding}(w_1) \\ \text{Embedding}(w_2) \\ \vdots \\ \text{Embedding}(w_L) \end{bmatrix}$$

This representation allows the text to be structured in a way that convolutional filters can process effectively.

Convolution Over Word Sequences A convolutional layer applies a set of filters over sliding windows of consecutive word vectors. A filter of size h is a matrix $\mathbf{W} \in \mathbb{R}^{h \times d}$. As it's illustrated in 4, is applied to each window of h words from the article, computing a scalar activation at each position i:

$$c_i = f(\langle \mathbf{W}, \mathbf{A}_{i:i+h-1} \rangle + b)$$

where:

- $\mathbf{A}_{i:i+h-1} \in \mathbb{R}^{h \times d}$ is the submatrix formed by the vectors $\mathbf{m}_i, \ldots, \mathbf{m}_{i+h-1}$,
- $\langle \cdot, \cdot \rangle$ denotes the convolution operation (often an element-wise multiplication followed by a sum),
- $b \in \mathbb{R}$ is a bias term,
- f is a non-linear activation function, we use ReLU: $f(x) = \max(0, x)$.

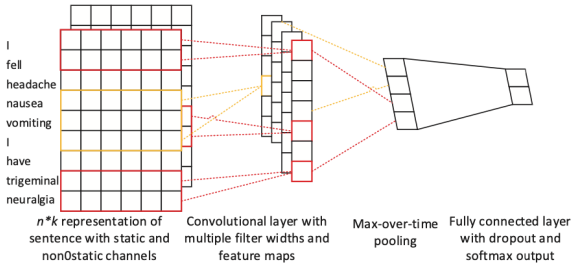


Figure 4: CNN for text classification (source : Cuili Yao and F [2016]

Each filter generates a feature map $\mathbf{c} = (c_1, c_2, \ldots, c_{L-h+1})$ indicating the strength of the pattern match at different positions.

Then the idea is to reduce dimensionality and "keep" only the most significant information from each feature map, we apply a ***max-pooling*** operation:

$$\hat{c} = \max\{c_1, c_2, \ldots, c_{L-h+1}\}$$

This operation keeps the strongest response from each filter, making the model invariant to the exact position of key patterns in the text.

Concatenation and Classification

In our implementation, the output of the convolutional layer is passed through a *global max-pooling* layer, which reduces each feature map to a single scalar value. Since we use a single convolutional layer with 128 filters, this results in a feature vector $\mathbf{z} \in \mathbb{R}^{128}$, where each element represents the maximum activation over time for one convolutional filter:

$$\mathbf{z}_j = \max\{c_1^{(j)}, c_2^{(j)}, \ldots, c_{L-h+1}^{(j)}\}, \quad j = 1, \ldots, 128$$

This feature vector is then passed through a fully connected dense layer of 128 neurons with ReLU activation, allowing the model to learn higher-level interactions between the features extracted by the filters.

To reduce overfitting, we apply dropout with a rate of 0.5 after the dense layer. Finally, the output layer consists of a single neuron with a sigmoid activation function, which produces a probability $\hat{y} \in [0, 1]$ indicating whether the input article is predicted to be fake or real:

$$\hat{y} = \sigma(\mathbf{w}^\top \mathbf{h} + b)$$

where $\mathbf{h} \in \mathbb{R}^{128}$ is the output from the dense layer, and σ is the sigmoid function defined by:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

This architecture is implemented using the Keras `Sequential` API, combining the embedding, convolutional, pooling, and dense layers into a single end-to-end trainable model.

Neural Net models Training

The models we presented are trained by minimizing the binary cross-entropy loss:

$$\mathcal{L}(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

where $y \in \{0, 1\}$ is the ground truth label. Optimization is typically performed using stochastic gradient descent (SGD) or the Adam algorithm, with backpropagation used to update both the convolutional filters and embedding weights.

- **And what about using an LLM for zero shot prediction ?**

We'll see in the next sections of the reports that LLM could be an excellent way of detecting false information, thanks to the great properties enabled by a specific architecture. (cf. section 5 : Appendix)

4 Results and Discussion

In this study, we designed an automated pipeline to systematically evaluate various combinations of feature extraction techniques (BoW, TF-IDF, BERT, Word2Vec) and classification models (Logistic Regression, Naive Bayes, SVM, Random Forest) for the task of fake news detection.

Each model was trained on the ISOT dataset, a benchmark dataset widely used in the literature for this task. Following training, we evaluated model performance on a test set drawn from the same ISOT dataset, allowing us to measure the ability of each model to learn from known data.

Beyond this initial evaluation, and in alignment with the goals of the original research paper, we also sought to assess the generalization capabilities of these models. Specifically, we tested the trained models without retraining on a completely different dataset: the Combined dataset, which merges data from multiple sources. This setup allowed us to evaluate whether the models could adapt to a distribution shift, which is crucial for real-world applications of fake news detection.

The results of these experiments are summarized in the table below:

Table 1: Résultats finaux de classification selon le vectoriseur et le modèle

Vectorizer	Model	Test Accuracy	New Data Accuracy
BoW	Logistic Regression	0.9963	0.5380
BoW	Naive Bayes	0.9441	0.5417
BoW	SVM	0.9951	0.5387
BoW	Random Forest	0.9980	0.6008
TF-IDF	Logistic Regression	0.9880	0.4949
TF-IDF	Naive Bayes	0.9288	0.5530
TF-IDF	SVM	0.9949	0.4666
TF-IDF	Random Forest	0.9978	0.6028
BERT	Logistic Regression	0.9886	0.5417
BERT	Naive Bayes	0.8994	0.6028
BERT	SVM	0.9937	0.5433
BERT	Random Forest	0.9588	0.5450
Word2Vec	Logistic Regression	0.9587	0.5244
Word2Vec	Naive Bayes	N/A	N/A
Word2Vec	SVM	0.9589	0.5274
Word2Vec	Random Forest	0.9584	0.5151

What is fundamental to observe in this table is that the models achieving the highest performance on the ISOT test set do not necessarily generalize best to the new dataset. For instance, combinations such as BoW or TF-IDF with powerful classifiers like Random Forest or SVM yield very high test accuracies (up to 99.8%), but their performance can drop significantly (to around 50–60%) on the Combined dataset. This suggests overfitting to the specific linguistic and structural patterns of the ISOT dataset.

On the other hand, some models such as BERT or Word2Vec combined with Random Forest or Naive Bayes exhibit more stable and higher generalization scores, notably BERT + Naive Bayes (0.6028) or BoW + Random Forest (0.6008). This indicates a better ability to handle linguistic variability and unseen textual patterns in the external dataset.

The moderate performance observed on the Combined dataset highlights the inherent difficulty of generalizing fake news detection models in real world scenarios. It underscores the need to design

robust models capable of adapting across diverse domains, sources, and writing styles.

In addition to traditional machine learning models, we also evaluated the performance of deep learning approaches using two architectures: LSTM combined with Word2Vec embeddings, and a CNN with an embedding layer trained from scratch.

Table 2: Résultats des modèles de deep learning

Vectorizer	Model	Test Accuracy	New Data Accuracy
Word2Vec	LSTM	0.9982	0.6632
Embedding Layer	CNN	0.9800	0.5729

The LSTM model with Word2Vec embeddings achieved a near-perfect accuracy of 99.8 on the ISOT test set, and more importantly, generalized best among all tested models, reaching 66.3 accuracy on the external Combined dataset. This result highlights the strong representational power of recurrent neural networks when paired with rich, pretrained semantic embeddings like Word2Vec.

The CNN model using a trainable embedding layer also performed well, with 98.0 accuracy on ISOT and 57.3 on the Combined dataset, confirming that convolutional architectures can effectively capture local patterns in text even when no pretrained embeddings are used. These observations suggest that deep learning models, particularly LSTMs with pretrained embeddings, are better suited for generalizing to unseen domains, making them promising candidates for real-world deployment in fake news detection systems.

To further assess the potential of generalization without any supervised fine-tuning, we experimented with a zero-shot classification approach using the facebook/bart-large-mnli model. This large language model is capable of inferring textual entailment and making predictions on new data based solely on natural language prompts, without seeing any examples during training.

Table 3: Résultat du modèle LLM en zero-shot (BART)

Model	New Data Accuracy
facebook/bart-large-mnli	0.7219

The BART-based model achieved 72.2 accuracy on the Combined dataset in a zero-shot setting, without any task-specific fine-tuning. This result illustrates the power of transfer learning in large language models: leveraging broad pre-trained language understanding, BART generalizes effectively across domains. Unlike traditional models that suffer from distribution shifts, LLMs like BART offer a robust alternative for low-resource or unseen scenarios.

5 Conclusion

This project demonstrates the effectiveness of NLP techniques in detecting misinformation. Classical models such as SVM and Logistic Regression perform well, but deep learning models like BART provide superior results. Future work could involve fine-tuning pre-trained models and exploring additional datasets.

This methodological proposal is motivated by the approach presented in the referenced article, where word frequency analysis is used to identify the terms that contribute most to the classification process.

The diagram in 5 illustrates a method designed to enhance the interpretability of a fake news detection system by leveraging the explanatory capabilities of a Large Language Model. The core motivation behind this approach is to understand the underlying factors contributing to the classification of a piece of information as false. While prior work often involves analyzing word frequency or importance scores to interpret model decisions, this method takes a step further by reusing the generative capabilities of the same model to provide a textual explanation for its classification.

In practice, once the model performs the classification task, it is subsequently prompted in its generative mode to justify its decision through natural language. This enables an interactive and inter-

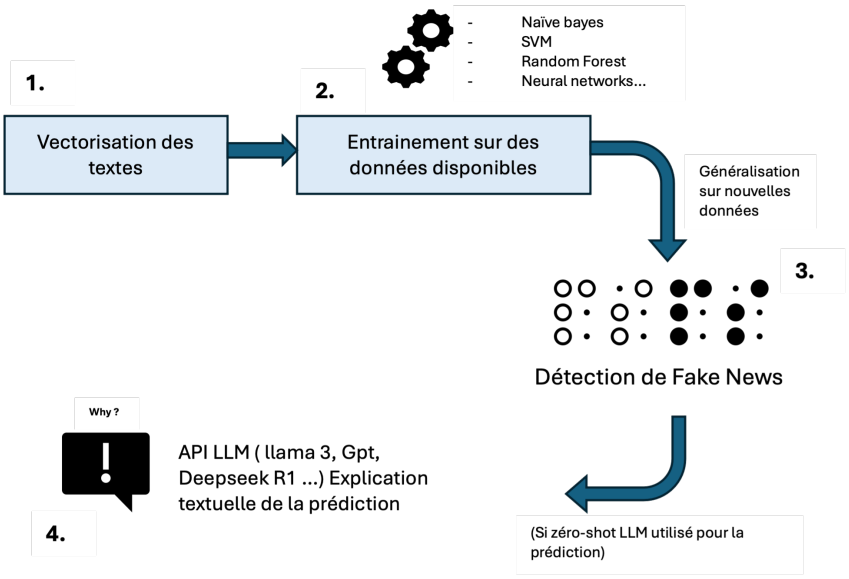


Figure 5: Method suggestion for fake news detection improving interpretability of the results

pretable framework where users can better understand how and why the model reached a particular conclusion. It also allows for human verification of the model's prediction, thereby increasing the transparency and reliability of the system.

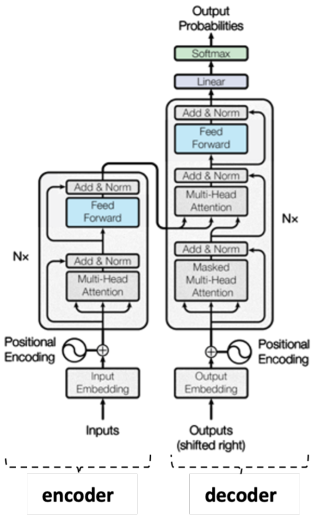
Appendix

1. Transformers introduction for Bert & LLMs

To briefly explain how it works, the Transformer follows an encoder-decoder architecture. The encoder transforms an input sequence into a continuous representation. A sequence refers to data ordered according to a convention (such as time, linguistic rules, etc.); in our case, this sequence is the prompt, which we construct using instructions and code. Each position corresponds to a token—typically a word from the prompt, although this can vary depending on the model. The decoder then uses this representation to generate an output sequence.

Each component (encoder and decoder) is made up of multiple identical stacked layers—six in the base model. These layers use multi-head self-attention mechanisms (which allow each position in the sequence to attend to all other positions) followed by fully connected neural networks. Multi-head attention enables the model to capture different sub-representations—that is, projections of the data into several distinct vector spaces simultaneously—and to establish dependencies between elements of the sequence.

In this architecture, a key-query-value system is implemented to represent sequence dependencies. Additionally, normalization is applied to ensure that model weights are computed appropriately. The absence of recurrence in this model helps avoid certain limitations typically associated with sequential processing.



Transformer architecture

Figure 6: Transformer architecture

Contribution of each member of the team

- Amine Razig : Contributed to the writing sections explaining BERT, CNNs, and the Random Forest. Took part in discussing results, creating tables. For the code, worked on the Zero-Shot prediction and LSTM and implemented classical models using BoW, TF-IDF, and BERT, as well as CNN. Participated in writing the introduction, related work and the conclusion.
- Mohamed Keteb : Contributed to the report by writing about the Word2Vec model, Logistic Regression, SVM, Bag of Words (BoW), TF-IDF, and LSTM. Participated in coding the LSTM model and implementing classical models with Word2Vec. Participated in writing the introduction, the related work and discussing results as well as the conclusion.

GitHub

Instructions to run the code are available in the README file of the following repository :

$https://github.com/arazig/Misinformation-detection$

References

Hadeer Ahmed, Issa Traore, and Sherif Saad. Detection of online fake news using n-gram analysis and machine learning techniques. In Issa Traore, Isaac Woungang, and Ahmed Awad, editors, *Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*, volume 10618 of *Lecture Notes in Computer Science*, pages 127–138. Springer International Publishing, 2017. doi: 10.1007/978-3-319-69155-8_9. URL https://doi.org/10.1007/978-3-319-69155-8_9.

Hadeer Ahmed, Issa Traore, and Sherif Saad. Detecting opinion spams and fake news using text classification. *Security and Privacy*, 1(1):e9, 2018. doi: 10.1002/spy2.9. URL https://doi.org/10.1002/spy2.9.

Ciara Blackledge and Amir Atapour-Abarghouei. Transforming fake news: Robust generalisable news classification using transformers. *CoRR*, abs/2109.09796, 2021. URL https://arxiv.org/abs/2109.09796.

Bo Jin Senior Member IEEE Li Guo Chao Li Wenjuan Cui Cuili Yao, Yue Qu and Lin F. A convolutional neural network model for online medical guidance. *IEEE*, 2016.

Akansha Gautam and Koteswar Rao Jerripothula. Sgg: Spinbot, grammarly and glove based fake news detection. In *2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM)*, pages 174–182. IEEE, Sep 2020. doi: 10.1109/BigMM50568.2020.00039.

Nathaniel Hoy and Theodora Koulouri. Exploring the generalisability of fake news detection models. *Brunel University Research Archive*, 2023. URL https://bura.brunel.ac.uk/bitstream/2438/25909/1/FullText.pdf.

Sawinder Kaur, Parteek Kumar, and Ponnurangam Kumaraguru. Automating fake news detection system using multi-level voting model. *Soft Computing*, 24(12):9049–9069, Jun 2020. doi: 10.1007/s00542-020-05634-3.

Junaed Younus Khan, Md Tawkat, Islam Khondaker, Sadia Afroz, Gias Uddin, and Anindya Iqbal. A benchmark study of machine learning models for online fake news detection. *2021*, 2021.

Karishnu Poddar, Geraldine Bessie Amali D., and K.S. Umadevi. Comparison of various machine learning models for accurate detection of fake news. In *2019 Innovations in Power and Advanced Computing Technologies (i-PACT)*, volume 1, pages 1–5. IEEE, Mar 2019. doi: 10.1109/i-PACT.2019.8891289.