From Text to Actuarial Modelling: The Role of NLP and LLMs in Cyber Risk Assessment

Hugo RAPIOR, Nexialog Consulting

6th European
Congress of Actuaries
www.eca2026.org

TABLE OF CONTENTS

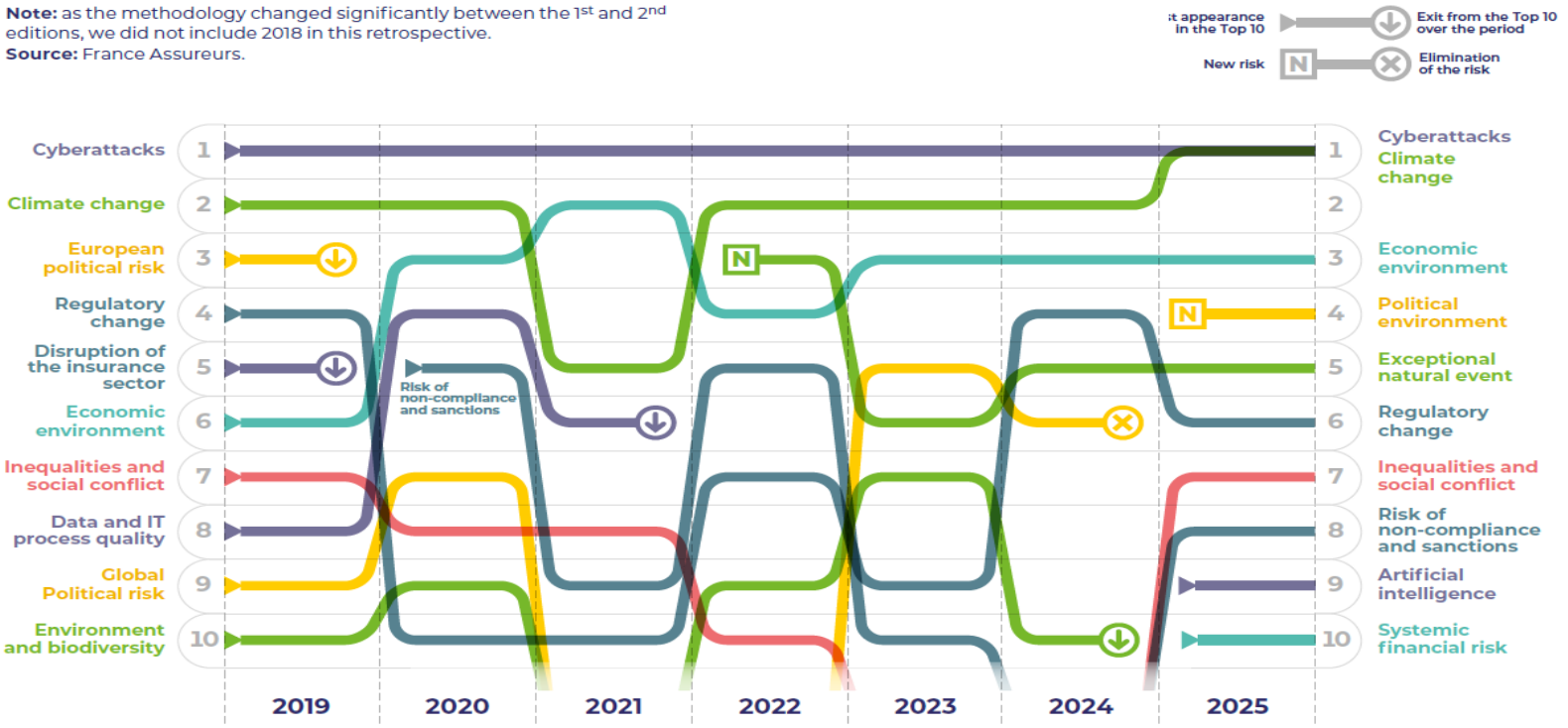
01

Context

Cyber Risk Context

High-stakes risks for the insurance sector

Note: as the methodology changed significantly between the 1st and 2nd editions, we did not include 2018 in this retrospective.
Source: France Assureurs.

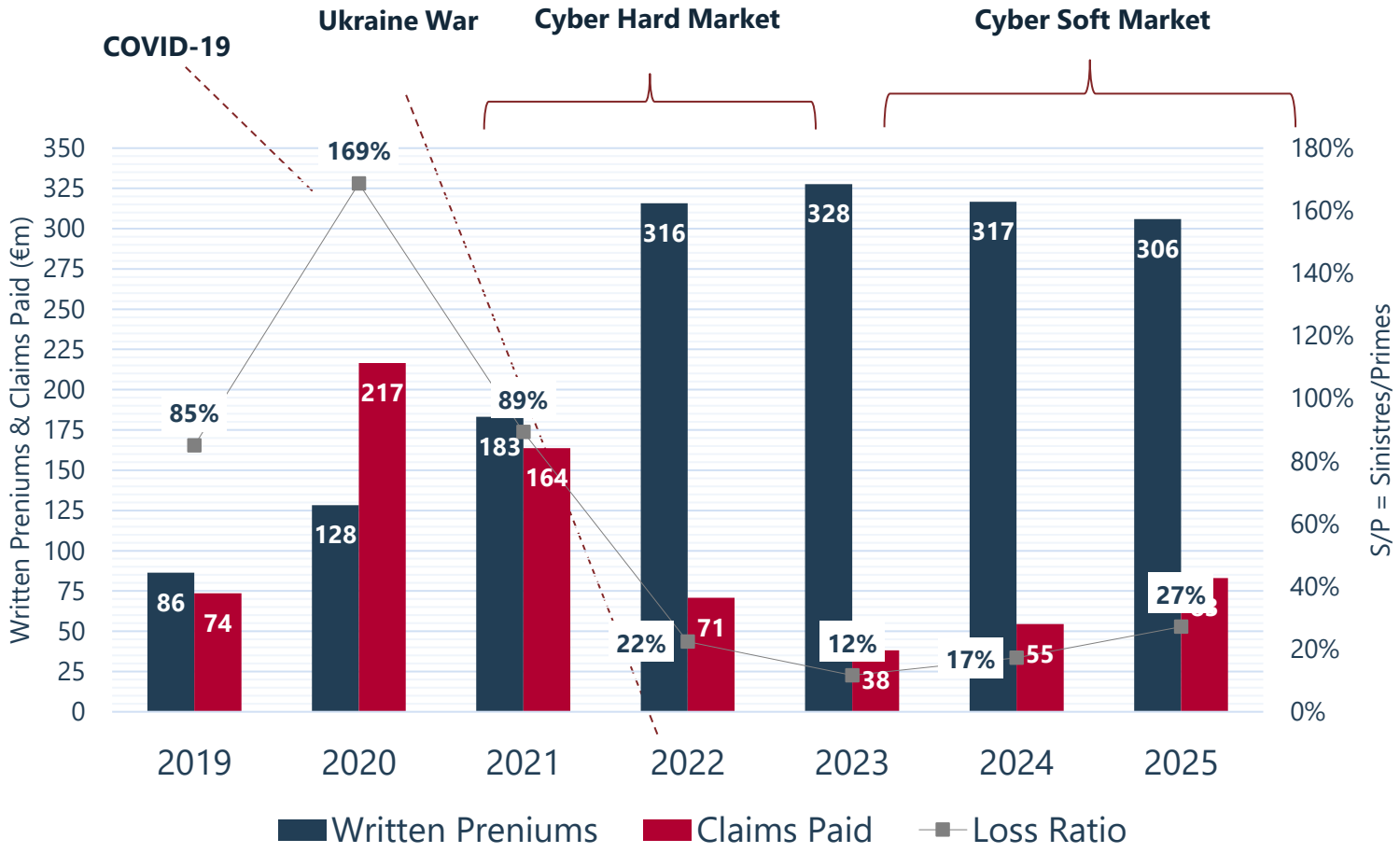


France Assureurs Map – 2019-2025 retrospective

Cyber Risk Context

A risk strongly linked to geopolitical conditions

- The different geopolitical situations have positively or negatively affected the performance of the Cyber insurance market



02

The PRC Database

The PRC Database

Presentation of the database

- Public database of data breaches in the United States
- Contains 74,000+ cyber incidents documented since 2005
- Allows tracking and analyzing data breaches by organization, sector, and attack type
- Academic and public reference for understanding data breach trends
- A variable indicative of severity: the "number_of_records"

Variable Type	Variables	Use in Modeling
Categorical qualitative	organization_type, organization_type_explanation, breach_type, breach_type_explanation	Embeddings or Co-variables
Textual	incident_details, information_affected, information_affected_explanation, impact_info_explanation	NLP → vectorization
Raw numeric	total_affected / number_of_records, residents_affected	Target variable, features
Temporal	breach_date, reported_date, end_breach_date, reporting_delay, exposure_duration	Date encoding

The PRC Database
An unstable structure over time

The PRC database variables changed in 2024, both in their construction and in the type of data collected.

Variable Type	PRC 2025 Variables	PRC 2019 Variables
Categorical qualitative	organization_type, breach_type	Typ_of_organization, Type_of_breach
Textual	incident_details, pdf_contents_cleaned, information_affected	Description_of_incident
Raw numeric	total_affected, residents_affected	Number_of_Records
Temporal	breach_date, reported_date, end_breach_date	Date_Made_Public, Year_of_Breach

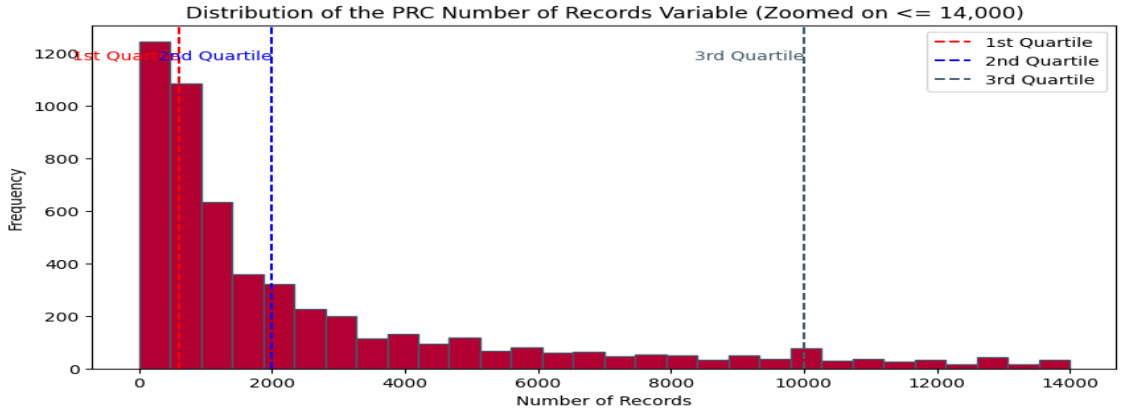
Direct consequence: the distribution of the 'number of records' variable (which connotes severity) also changed radically.

Statistic	Severity PRC 2025 (2005–2025)	Severity PRC 2025 (2005–2019)	Severity PRC 2019 (2005–2019)
# claims	31 217	7 757	6 822
Mean	434 789	781 721	1 522 632
Std. dev.	15 727 315	25 836 170	41 960 690
Min	1	1	1
q25%	121	13	613
q50%	1 302	517	2 000
q75%	8 661	3 532	10 000
q90%	53 828	23 382	63 000
q95%	182 504	80 000	259 123
Max	2 000 000 000	2 000 000 000	3 000 000 000

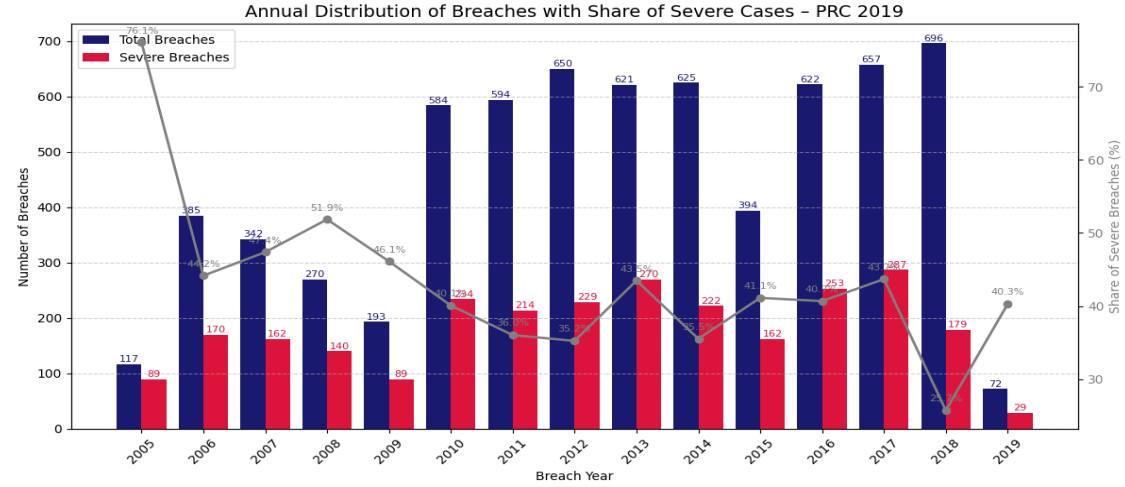
The PRC Database

Visualization of the "severity" distribution change

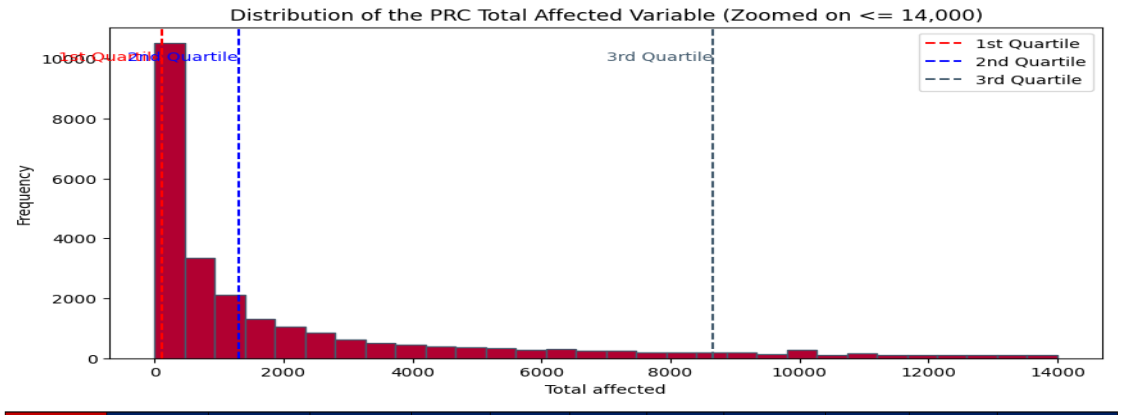
PRC 2019



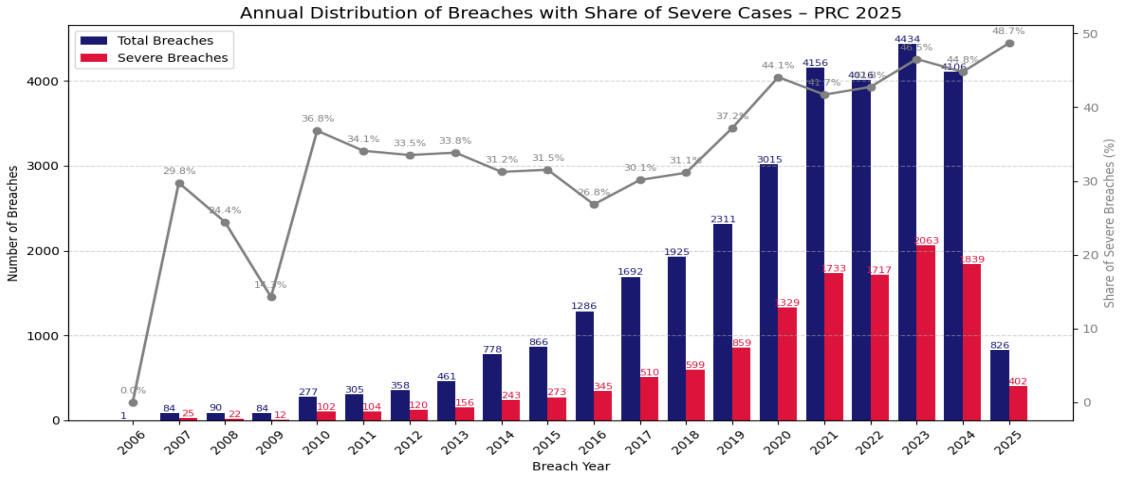
Statistics	count	mean	std	min	25%	50%	60%	75%	90%	95%	max
Value	6,822	1,522,632	41,960,690	1	613	2,000	3,386	10,000	63,000	259,123	3,000,000,000



PRC 2025



Statistics	count	mean	std	min	25%	50%	60%	75%	90%	95%	max
Value	31,217	434,789	15,727,315	1	121	1,302	2,525	8,661	53,828	182,504	2,000,000,000



- 03

Extreme Value Theory

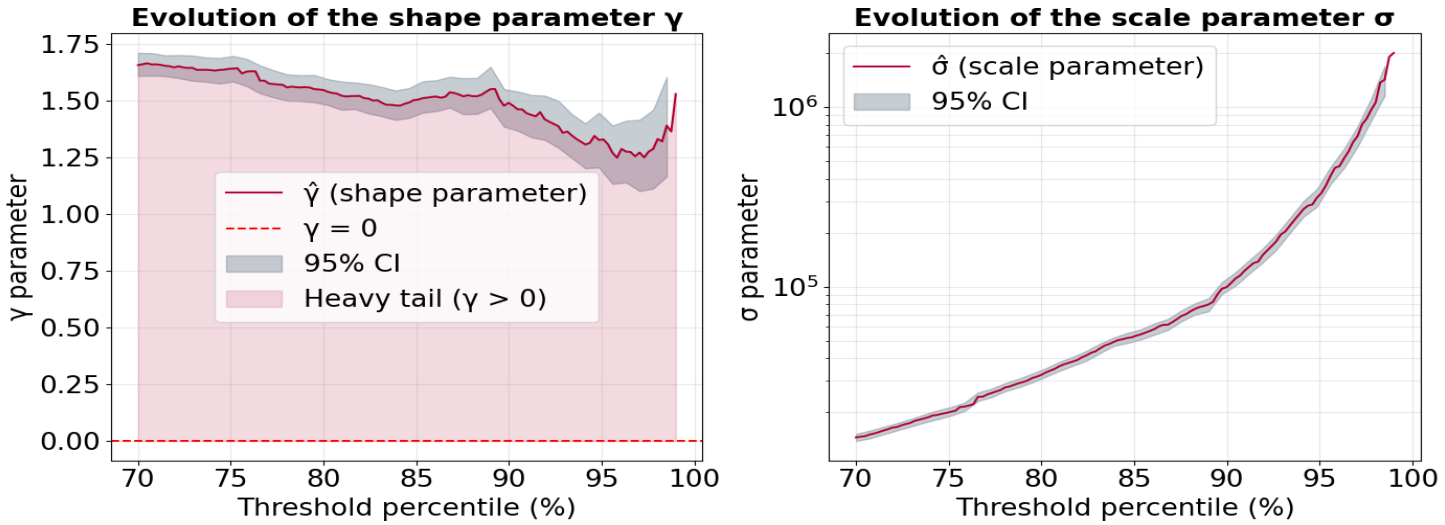
A bit of Extreme Value Theory

Peaks Over Threshold (POT) approach to determine our GPD

- Peaks Over Threshold (POT) approach and Generalized Pareto Distribution: Pickands–Balkema–de Haan theorem

Objective: calibrate the parameters of a Generalized Pareto Distribution on excesses (the location parameter) from a given threshold

The "optimal" severe threshold is considered reached when the shape parameter stops evolving



A bit of Extreme Value Theory

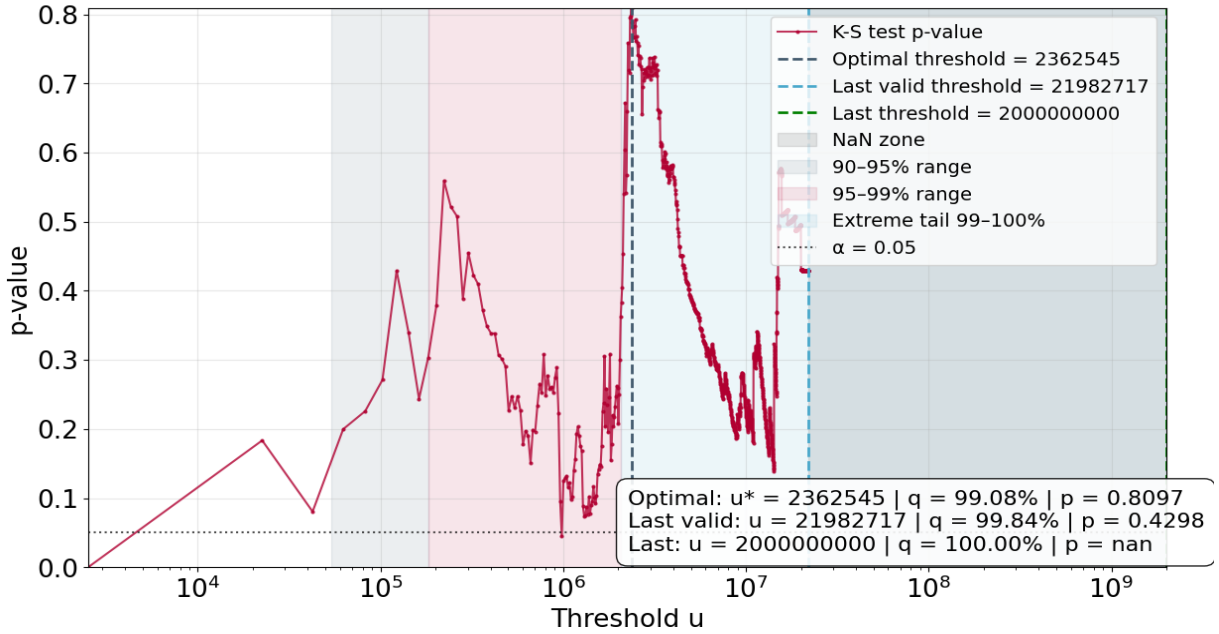
Definition of an optimal "severe" threshold

- The severity distribution of the database is excessively attritional, even more so than in the 2019 version → Need to better define a "severe" threshold

Idea: Calibrate a severe threshold such that the resulting "likelihood" would be maximized

Approach used: Kolmogorov-Smirnov test

- We test the goodness-of-fit between simulated data from calculated parameters and the empirical distribution H0
- The higher the p-value, the more confident we can be in keeping the threshold as the severe separation threshold
- We see a p-value maximization for a threshold of ~2.3M, equivalent to a 99th percentile



Percentile	n	$\hat{\gamma}$	95% CI (γ)	$\hat{\sigma}$	95% CI (σ)
95th	1,561	1.3247	[1.2035 ; 1.4503]	326,668	[291,961 ; 365,415]
99th	313	1.5307	[1.2671 ; 1.7671]	2,005,313	[1,658,467 ; 2,498,293]

- 04

Introduction to NLP

Introduction to NLP

Text Pre-processing

Objective: prepare incident reports for automated language analysis (NLP).

Main steps:

1. **Cleaning**
   - Remove tags, URLs, unnecessary digits, convert to lowercase
2. **Tokenization**
   - Split text into words or expressions
3. **Stopword Removal**
   - Remove frequent words with no analytical value ("the", "and", "of")
4. **Lemmatization**
   - Reduce words to their base form
5. **Vocabulary Normalization**
   - Standardize technical or abbreviated terms
6. **Bigram and trigram generation**
   - "social security number" → "social_security_number"

Examples:

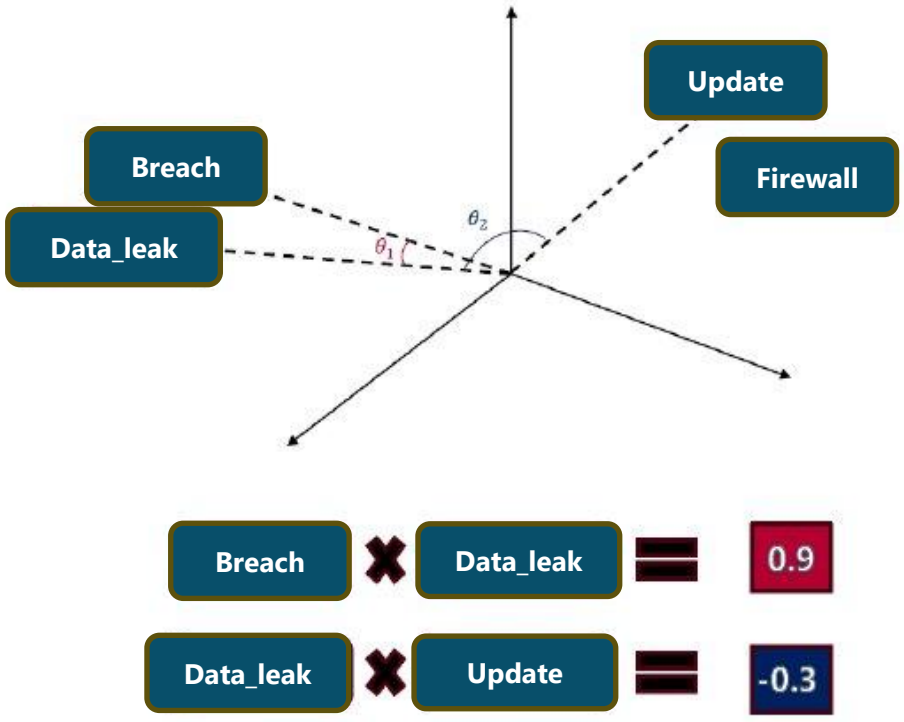
- "HTTP://YAHOO.COM" → "yahoo.com", "PASSWORDS" → "passwords"
- "Hackers accessed user data" → ["hackers", "accessed", "user", "data"]
- ["the", "breach", "of", "the", "system"] → ["breach", "system"]
- "hacked", "hacking" → "hack"
- "pwd" → "password", "info leak" → "information leak"
- "social_security_number", "district_attorney"

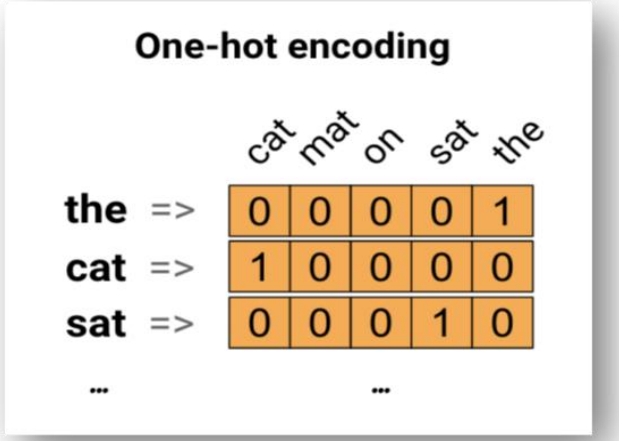
Introduction to NLP

Textual data pre-processing (Encoding) and Embedding

Once we have these "tokens", they must be converted into a format that the machine can understand.

- Text in itself cannot be understood; it must be represented as numbers.
- These encodings do not allow understanding more complex relationships between words. For that, a process called **embedding** is then used.





- **Embedding** allows representing words from the cyber vocabulary as vectors.
- Words that are **semantically close** (same attack type, same context) are **close in the vector space**, measured by **cosine similarity.**

$$\cos(\theta) = \frac{\vec{u} \cdot \vec{v}}{|| \vec{u} ||\times|| \vec{v} ||}$$

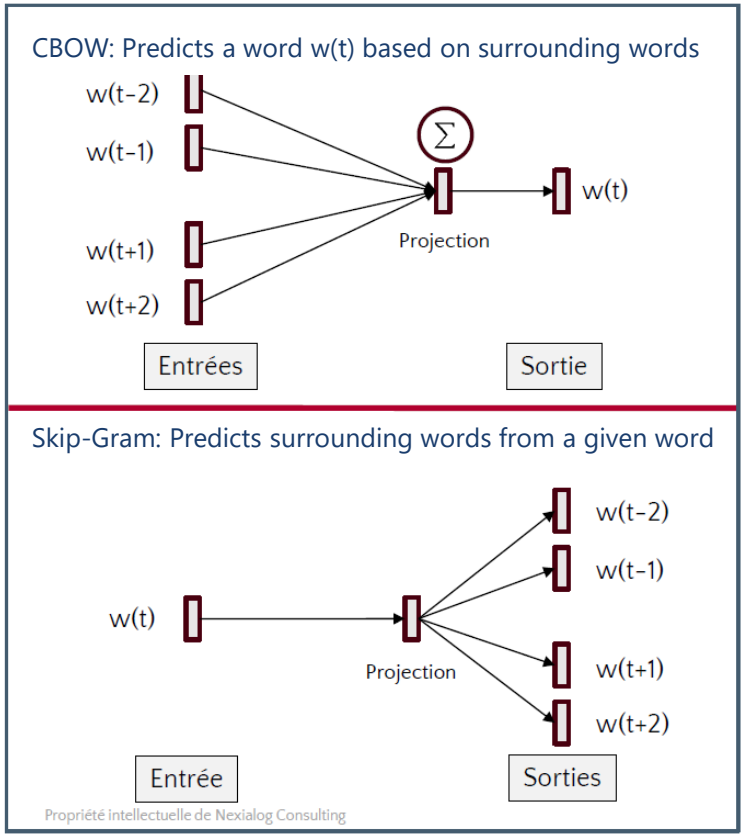
- Here, the angle between "**Breach**" and "**Data_leak**" is small → cos θ ≈ 0.9 (same lexical field: exfiltration, data loss, compromise)
- The angle between "**Data_leak**" and "**Update**" is large → cos θ ≈ –0.3 (an "update" is a preventive or corrective action, not an attack)

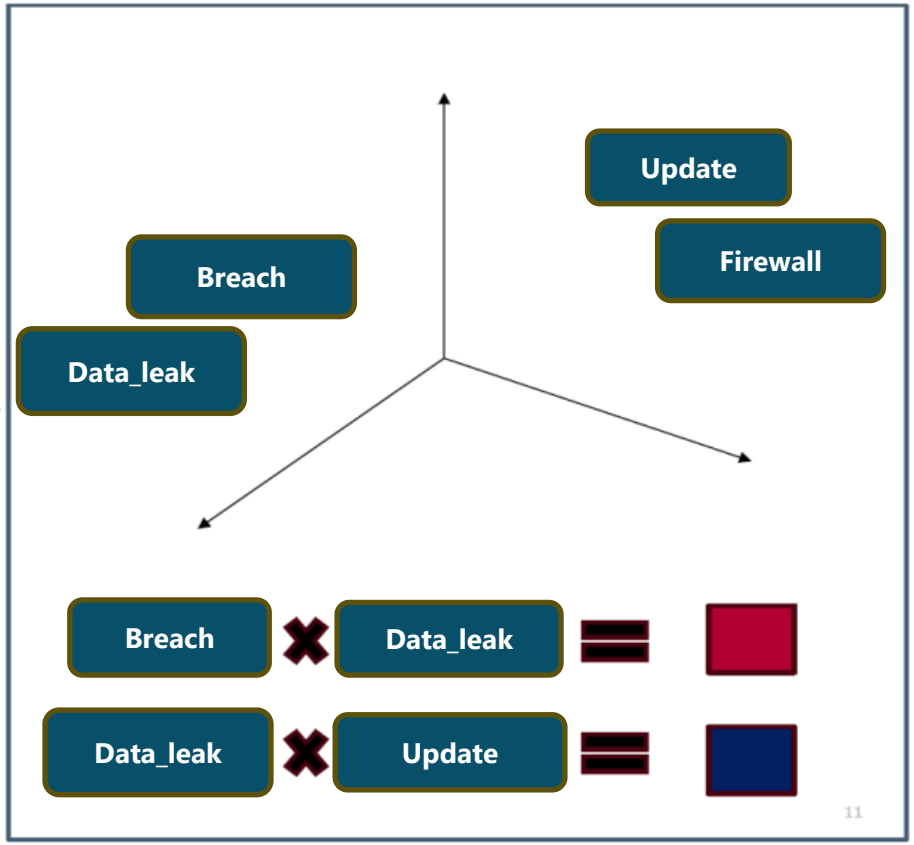
Introduction to NLP

Word2Vec (2013)

Objective: Learn word representations from large text corpora (e.g., incident reports, cyber press articles).

Techniques used: neural networks, frequency-based models (Word2Vec, FastText, BERT…).

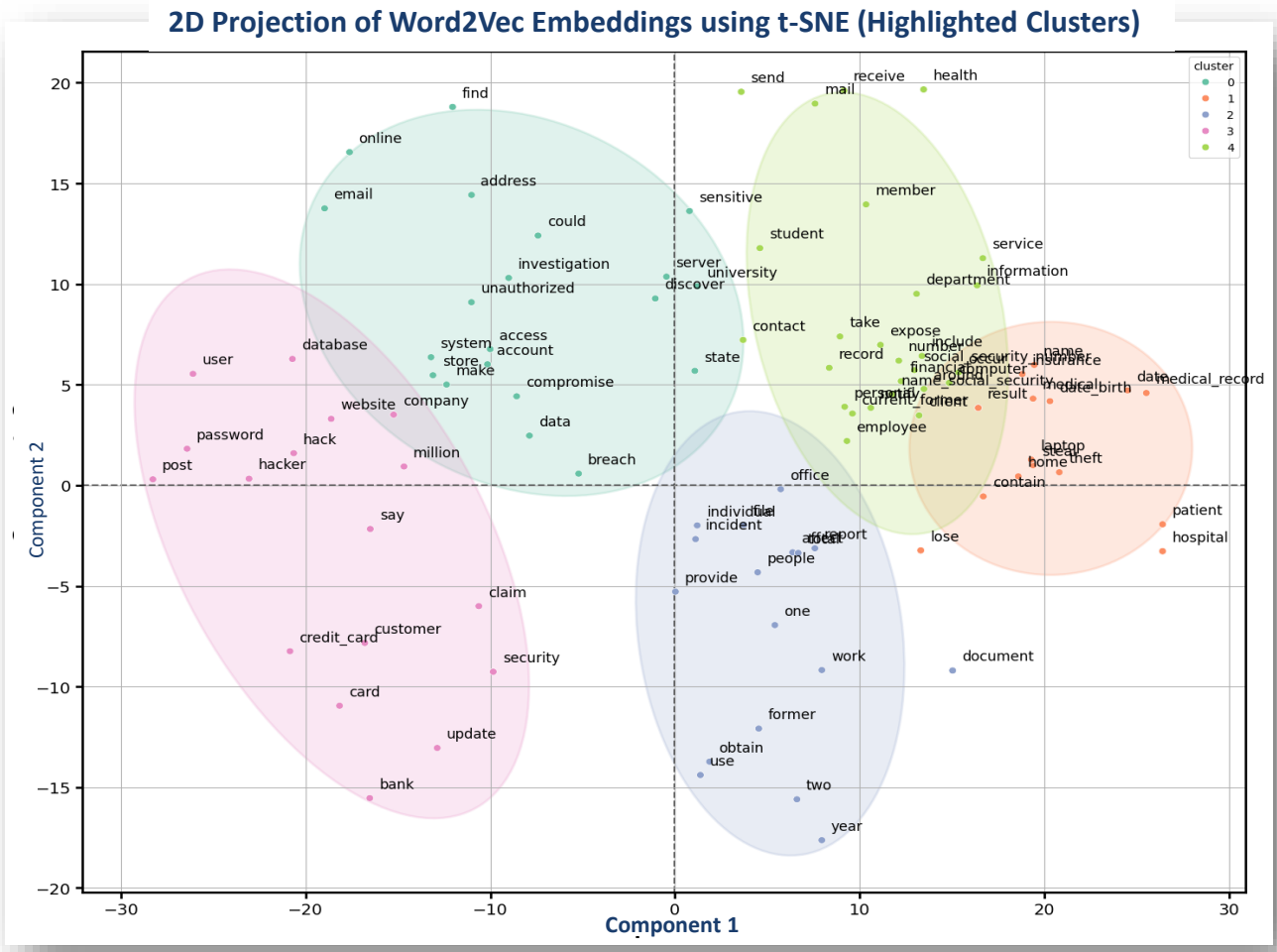




Introduction to NLP

Word2Vec (2013): results on the PRC 2025 database

- Important: This is a projection. The vectors produced by the algorithm have a much higher number of dimensions, to capture complex relationships between words.



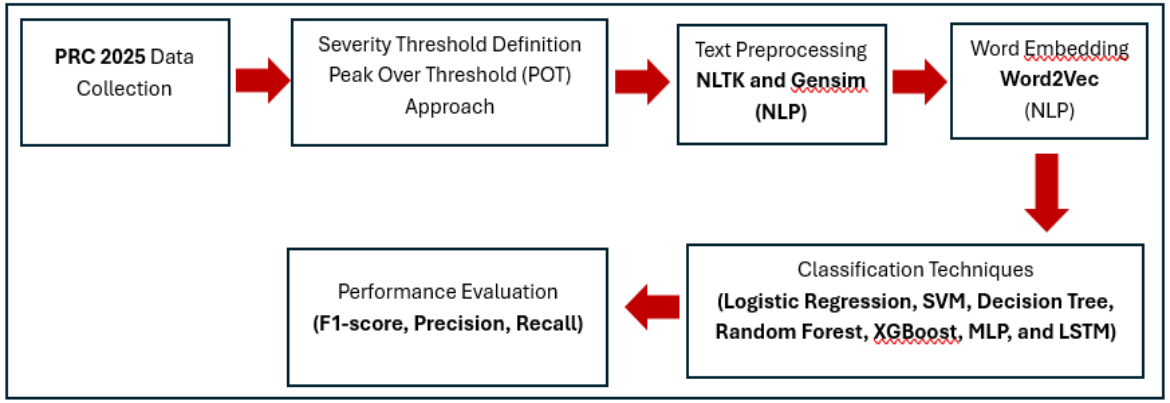
- 05

Deep Learning

Deep Learning

General Methodology

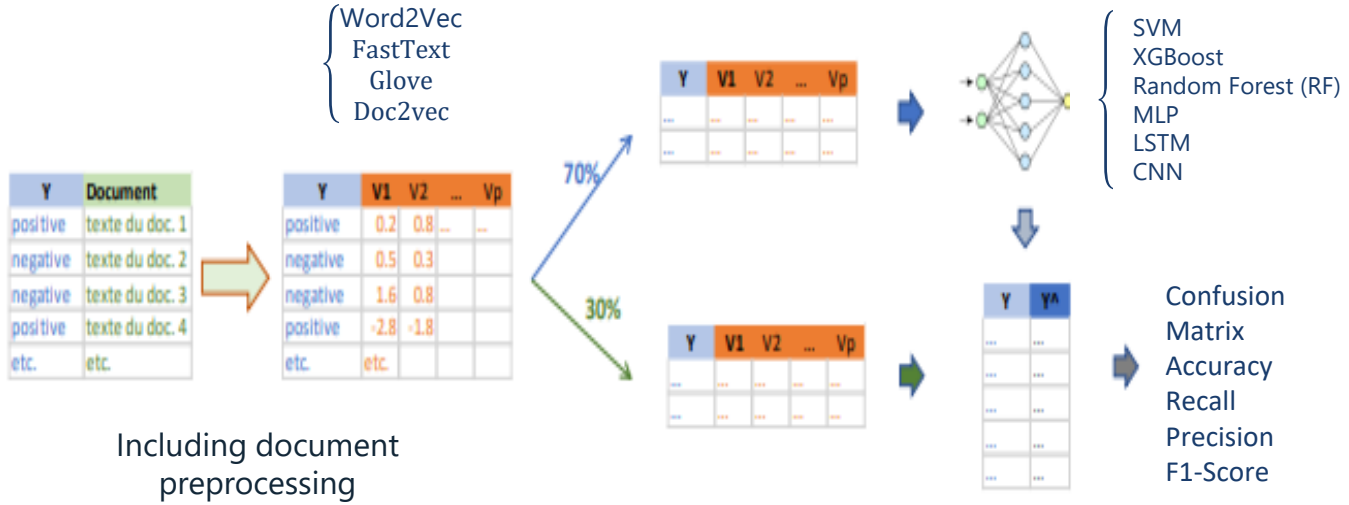
Once embeddings are created, **Machine Learning algorithms** can be applied (incident classification, threat detection, report clustering, etc.) on these vectors.



+

REGULAR EXPRESSIONS (RegEx)

Improve prediction of severe incidents



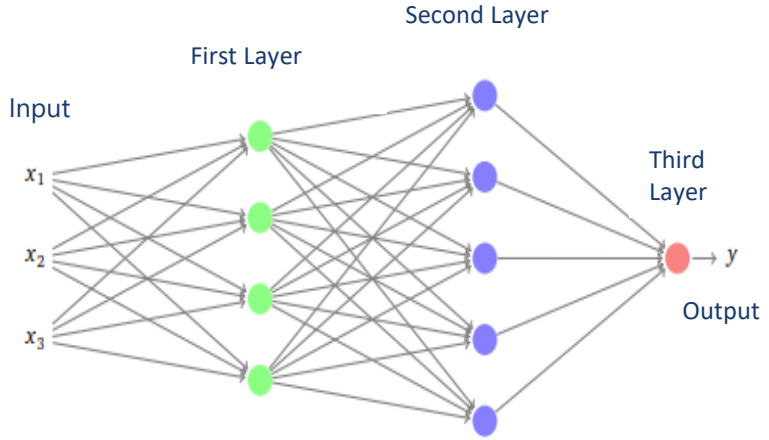
Including document preprocessing

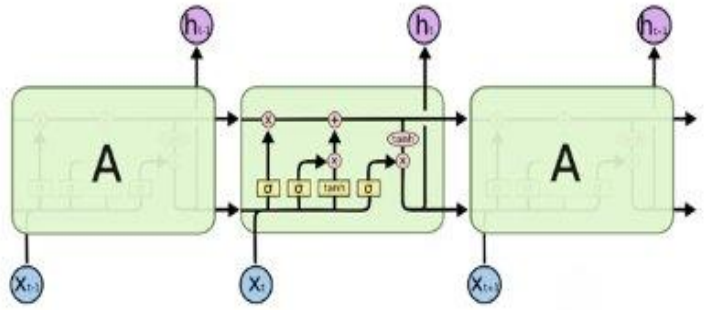
Deep Learning

LSTM Networks (Long Short-Term Memory)

1. MLP – Multi-Layer Perceptron

Key idea: mimics a biological neuron to learn relationships between input data and output.
- **Inputs**: x_1, x_2, ..., x_n, each associated with a weight w plus a bias.
- **Activation**: non-linear functions (sigmoid, ReLU, tanh…).
- **Architecture**:
  - **Input layer:** receives the data (e.g. text, vectors).
  - **Hidden layers:** learn complex, non-linear representations.
  - **Output layer:** produces the final prediction ŷ (severe vs. attritional).





Simplified structure of an LSTM cell (Olah 2015)

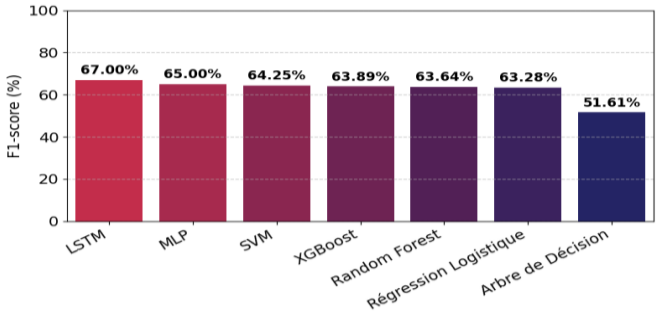
2. LSTM – Long Short-Term Memory: a solution to the memory problem

Key idea: solves the problem of long-term dependencies (vanishing gradient) in RNNs.
- **Standard RNN**: each output depends on the current input and the previous state, but forgets long-term information.
- **Improved LSTM (Hochreiter & Schmidhuber, 1997)**:
  - Introduces a **memory cell** c that carries information.
  - Uses 3 gates:
    - **Forget gate:** decides what to forget.
    - **Input gate:** decides what to add.
    - **Output gate:** decides what to transmit.Advantage: learns long-term context without information loss.
- **Advantage**: learns long-term context without information loss.

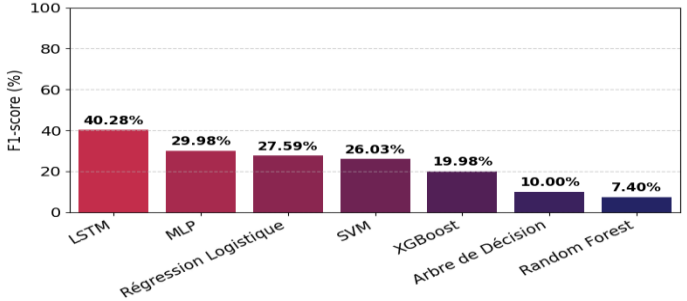
Deep Learning

MLP and LSTM Results (Long Short-Term Memory)

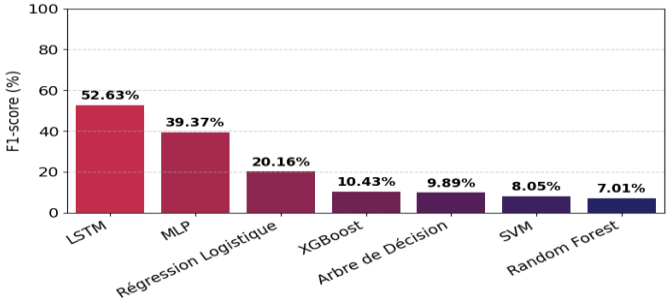
Analysis of results at the 60th percentile severity threshold



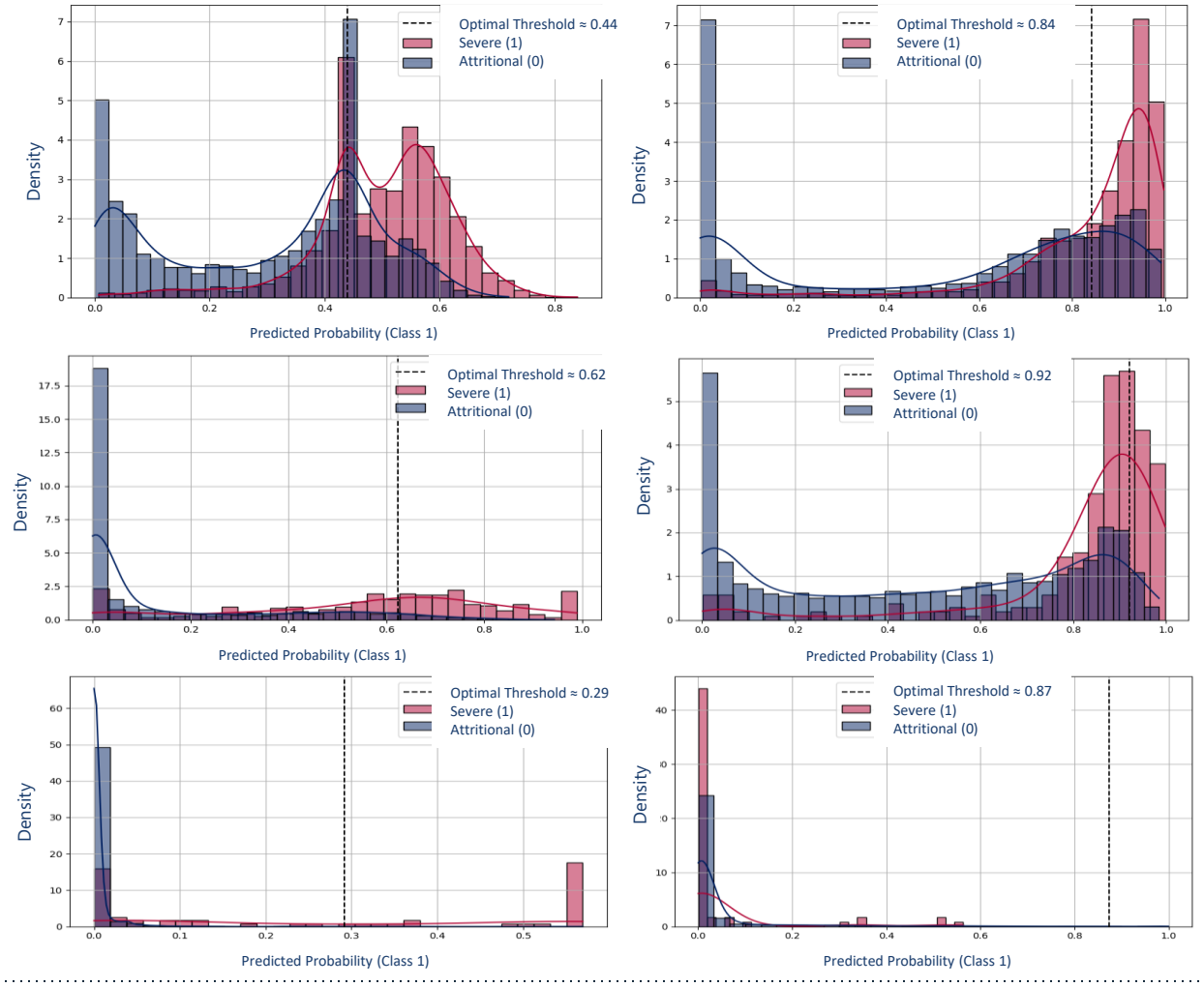
Analysis of results at the 95th percentile severity threshold



Analysis of results at the 99th percentile severity threshold



LSTM Results | **MLP Results**



06

Generative AI for Prediction

Generative AI for Prediction

The basic intuition: bootstrap

Idea: revisit the concept of Bootstrap

Bootstrap: for a linear functional (such as the mean), the bootstrapped estimator converges asymptotically without bias toward the true mean. For the mean, the variance of the bootstrapped estimator converges toward the empirical variance, which guarantees asymptotic unbiased convergence.

In plain terms: For a data sample $x_1, x_2, \ldots, x_n$, if the empirical mean is 70 —
if we bootstrap this sample a sufficient number of times and average the results, we will also converge toward 70.

Is this still true for text?

Intuitively, a LLM (ChatGPT, etc.) could be seen as a very large Bootstrap system, since it reuses words from sources it was trained on.

"Is the property of asymptotic convergence without bias of Bootstrap preserved when using LLM-generated text?"

Generative AI for Prediction

Method for generating claim descriptions

Idea: Define an approach (two in practice) to correctly simulate cyber incident descriptions

Two possibilities: let the LLM do it entirely, or guide it a bit (non-parametric vs. parametric)

Common Approach

25 examples each of severe and attritional claims from the PRC database are provided

Non-parametric

The LLM is asked to generate severe claim descriptions, without any additional constraint

Parametric

1) Key variable realizations (organization type, breach type, …) are simulated in advance
2) The LLM is asked to generate severe claim descriptions, considering simulated values for each key variable

In both approaches, many iterations were required to converge to a prompt producing satisfactory results (prompt engineering was necessary)

Generative AI for Prediction

Examples of generated claims

Real Incidents

{"organization_type": "BSR",

"breach_type": "CARD",

"incident_details": "The Wisconsin Department of Justice reported on December 19, 2013, that Target Corp. experienced a data breach affecting approximately 40 million customers. The breach occurred between November 27 and December 15, 2013, involving the theft of information from credit and debit cards, including customer names, card numbers, expiration dates, and CVV codes.",

"niveau_gravite_99": "grave"}

{"organization_type": "BSR",

"breach_type": "UNKN",

"incident_details": "The Indiana Office of the Attorney General reported a data breach involving The Neiman Marcus Group on September 30, 2021. The breach occurred on May 2, 2020, affecting a total of 4,354,346 individuals, including 31,992 residents.",

"niveau_gravite_99": "grave"}

Generated Incidents

{"organization_type": "BSO",

"breach_type": "HACK",

"incident_details": "The California Office of the Attorney General reported that a national online services provider experienced a network intrusion attributed to ransomware between April 3 and April 8, 2024. Forensic review indicates unauthorized access to systems containing names, Social Security numbers, and authentication data for approximately 6.1 million individuals across multiple states. The actor exfiltrated files before encrypting servers.",

"niveau_gravite_99": "grave"}

{"organization_type": "MED",

"breach_type": "HACK",

"incident_details": "The U.S. Department of Health and Human Services reported that a multi-state clinic network was attacked through a supply-chain exploit on May 9, 2024. Protected health information for 2.9 million patients, including medical histories, was compromised.",

"niveau_gravite_99": "grave"}

Generative AI for Prediction

Evaluation metrics for 'bias' and 'variance'

1. Jensen–Shannon Divergence (JSD)

- Objective: compare probability distributions between real and simulated data
- Evaluates the similarity between: distribution of categorical variables (e.g. organization_type, breach_type or vocabulary), generated vocabulary vs. original text.
- Interpretation: close to 0 → similar distributions; close to 1 → strong divergence

$$JSD(P_{real}, P_{syn}) = \frac{1}{2} D_{KL}(P_{real} \,||\, M) + \frac{1}{2} D_{KL}(P_{syn} \,||\, M)$$

with $M = \frac{1}{2}(P_{real} + P_{syn})$.

2. Center Bias (1 – cos)

- Objective: measure the global shift between the center of real and simulated embeddings.
- Interpretation: 0.00–0.05 → very close; 0.05–0.10 → slight shift; >0.10 → notable bias (generated content 'off the mark')

$$Biais_{center} = 1 - cos(\mu_{real}, \mu_{syn})$$

μ= centroid of word vectors.

3. MMD RBF (Maximum Mean Discrepancy – Radial Basis Function)

- Objective: compare the full shape of embedding distributions (variability, multi-modality) and capture shape discrepancies that a simple center cannot detect.
- Interpretation: $MMD^2 \approx 0$ → very close distributions; 0.10–0.20 → moderate gap; >0.20 → perceptible semantic bias

$$k(x, y) = e^{-\gamma \|x-y\|^2},$$

$$MMD^2 = E[k(x, x')] + E[k(y, y')] - 2E[k(x, y)]$$

Generative AI for Prediction

Bias evaluation according to different prompt approaches

After generating 400 severe claims according to the different approaches presented above, bias measures are evaluated by comparison with the real database of 313 severe claims:

Evaluation Metrics / Prompt Techniques	JSD (organization_type)	JSD (breach_type)	Center Bias	Lexical JSD	MMD RBF
Non-parametric generation	0.021	0.016	0.0780	0.753	0.2364
Parametric generation	**0.004**	**0.004**	**0.0453**	**0.692**	**0.1635**

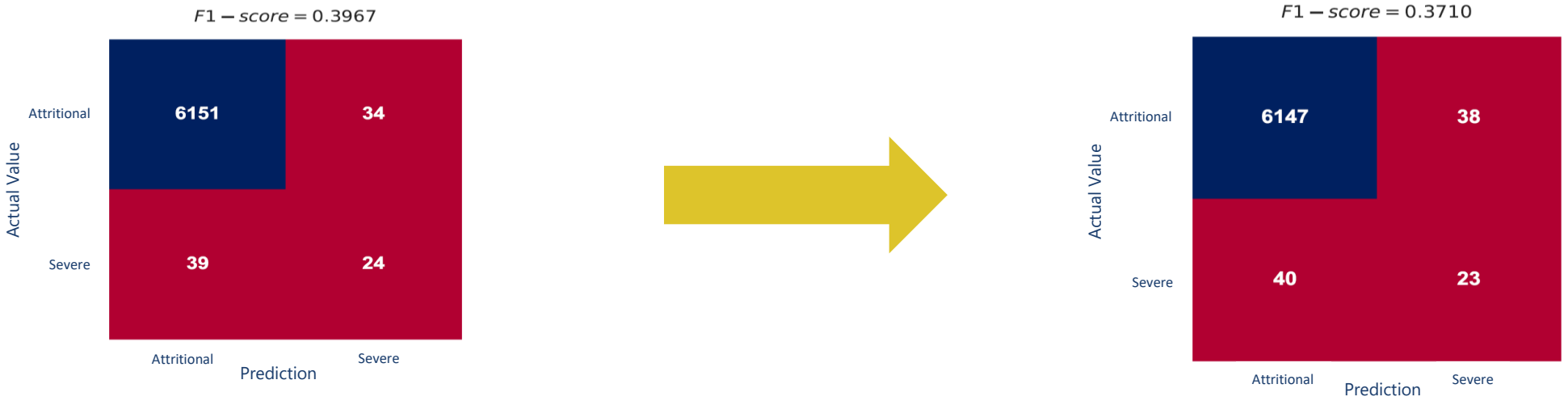
- Results show that **parametric approaches** significantly reduce all biases, reflecting **better semantic and structural similarity** with real claims.
- Integrating statistical distributions into the prompt therefore improves the **fidelity and consistency** of the generated synthetic data.

Generative AI for Prediction

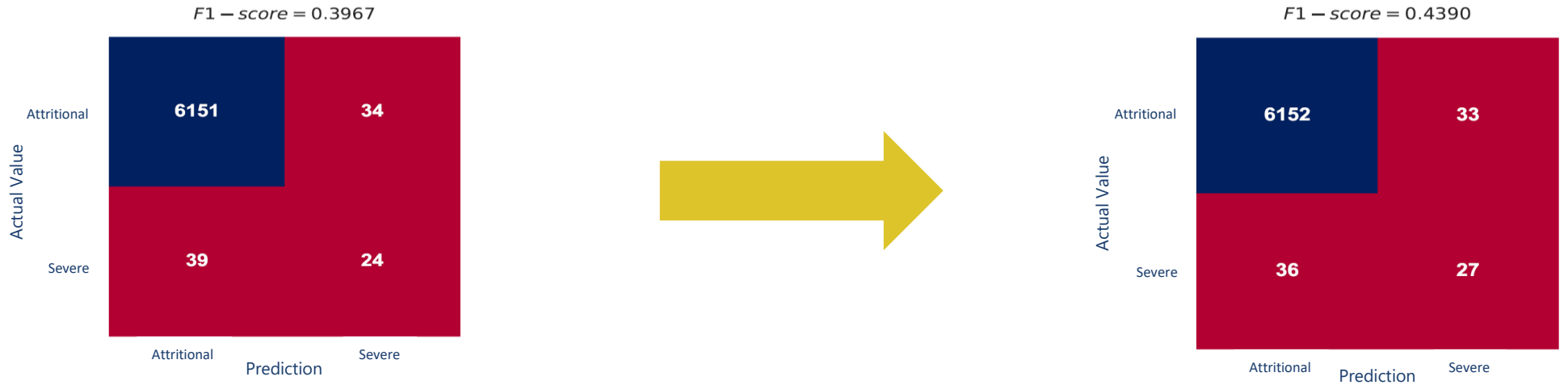
Results of adding synthetic data to the optimal MLP model

at the 99th percentile severity threshold

Result after adding 500 attritional claims generated using a parametric approach



Result after adding 5,000 severe claims generated using a parametric approach



- 07

RegEx Analysis

Regular Expressions

Reintroduction of numbers for prediction improvement

Motivation: addressing the limitations of Deep Learning

- Neural networks struggle to interpret numerical expressions in text (e.g., "27,000 employees", "315 patients").
- Embeddings (Word2Vec) separate numbers from words, losing their semantic link.
- Learning remains dependent on examples seen during training, limiting generalization to new values.

Solution: combine the power of LSTM with complementary analysis using regular expressions (RegEx) to capture number × word patterns.

Principle of the hybrid approach

- Final decision = Neural prediction (LSTM) OR RegEx prediction
- The RegEx model extracts and aggregates number × word combinations (e.g.: "20,787 individuals").
- A RegEx score is computed for each description.

$$\text{Score}_{\text{REGEX}}(D) = \sum_{u=1}^{n} N_u \cdot 1_{M_u \in V}$$

- The incident is classified as severe if

$$\text{Score}_{\text{REGEX}}(D) > \text{Seuil}_{\text{REGEX}}(k) \begin{cases} 2\,521 \; pour \; k = 60\% \\ 181\,820 \; pour \; k = 95\% \\ 2\,068\,450 \text{ pour k} = 99\% \end{cases}$$

where $k \in \{60,95,99\}$ corresponds to the severity percentiles.

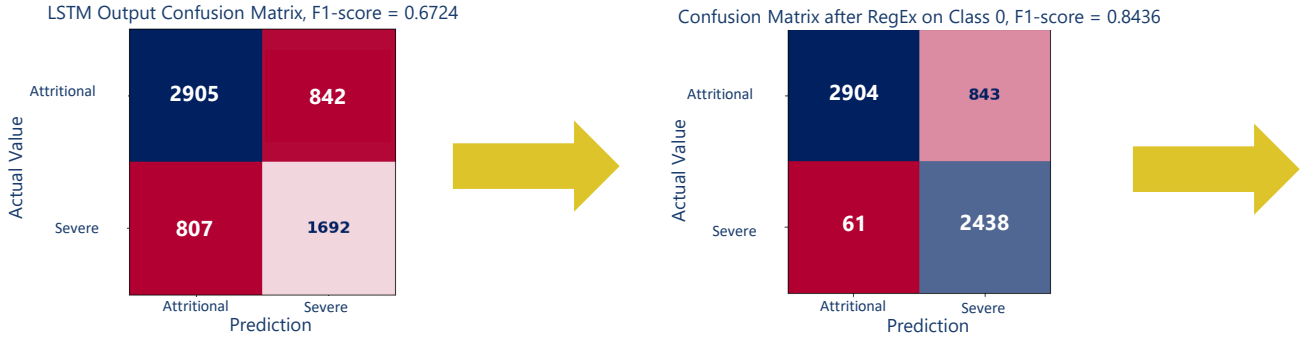
Operational methodology (for each description D and severity level k)

Step	Description	Output
1. Neural prediction	neural_prediction ← model.predict(D)	Probability from the LSTM
2. RegEx analysis	If D contains a "number × word" pattern: compute $\text{Score}_{\text{REGEX}}$	Binary RegEx prediction
3. Final decision	final_prediction = neural_prediction OR regex_prediction	Final classification

Regular Expressions

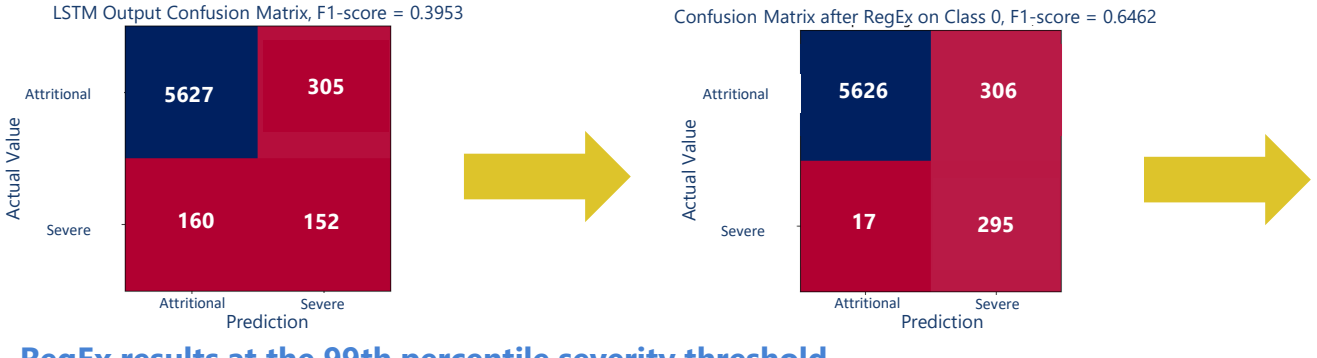
Results

RegEx results at the 60th percentile severity threshold



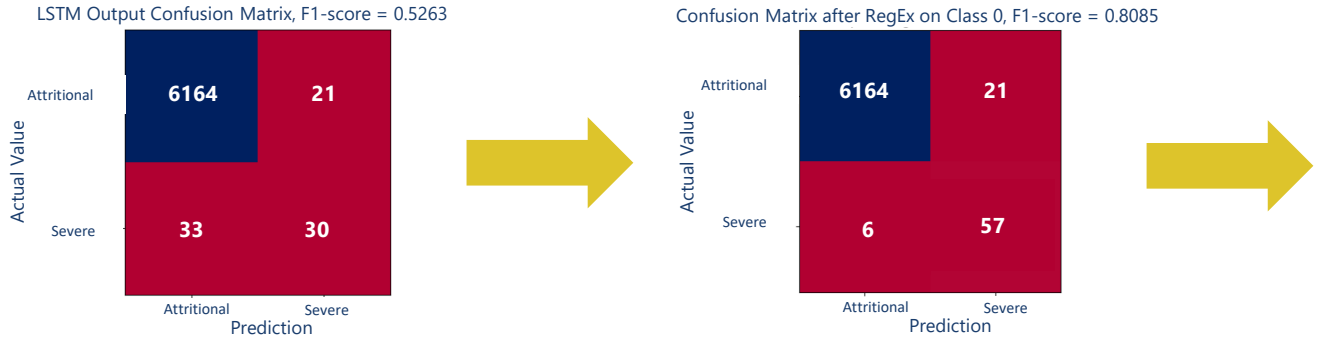
$$\Delta F_1 = F_1^{hybride} - F_1^{LSTM} = 0,3069$$

RegEx results at the 95th percentile severity threshold



$$\Delta F_1 = F_1^{hybride} - F_1^{LSTM} = 0,5461$$

RegEx results at the 99th percentile severity threshold



$$\Delta F_1 = F_1^{hybride} - F_1^{LSTM} = 0,2951$$

Conclusion

- Cyber risk is constrained by data scarcity and biases
- NLP enables exploiting text to predict claim severity
- The hybrid LSTM + RegEx approach significantly improves the detection of severe claims
- Generative AI makes it possible to simulate and enrich data, improving the detection of severe claims
- The challenge now is to combine statistics, NLP, and generative AI for more accurate, explainable, and dynamic models
- **However, simulated claims are easily differeciable by MIA (Membership Inference Attack) -> Need for adjustments to be GDPR compliant in case of real life data**