SysCRED_Documentation

🔬 SysCRED - Documentation Complète

Système Neuro-Symbolique de Vérification de Crédibilité

> **Version:** 2.0
> **Auteur:** Dominique S. Loyer
> **Citation Key:** `loyerModelingHybridSystem2025`
> **DOI:** 10.5281/zenodo.17943226
> **Dernière mise à jour:** Janvier 2026

📋 Table des Matières

Vue d'ensemble

Qu'est-ce que SysCRED?

SysCRED (System for CREdibility Detection) est un **système hybride neuro-symbolique** conçu pour évaluer automatiquement la crédibilité des informations en ligne. Il combine:

- **Approche symbolique** (règles explicites, transparentes et explicables)
- **Approche neuronale** (modèles NLP pour sentiment, biais, entités)
- **Ontologie OWL** (traçabilité et raisonnement sémantique)

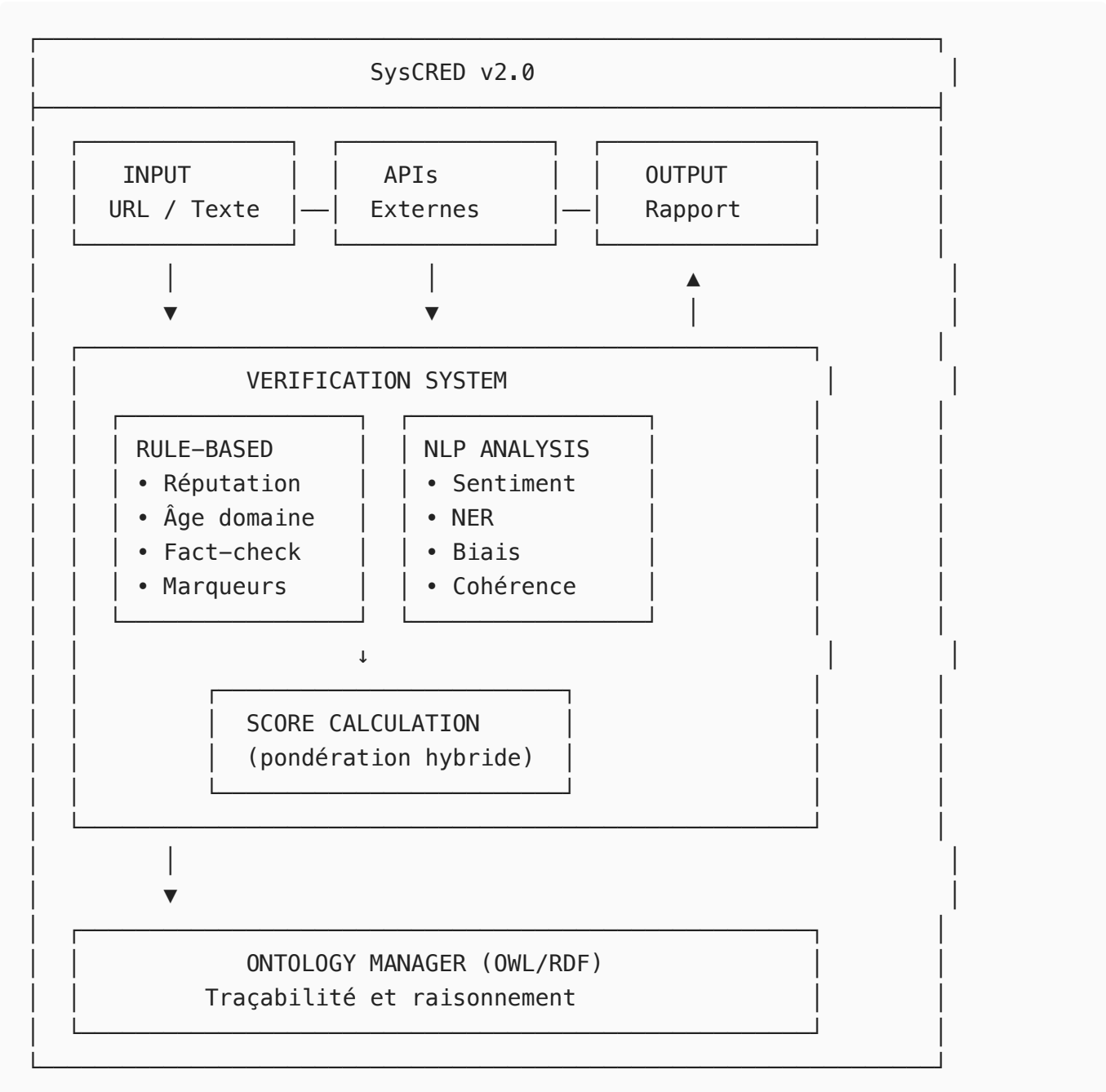
Philosophie du projet

Le système est conçu comme **prototype de recherche doctorale** avec ces principes:

1. **Explicabilité (xAI)**: Chaque décision peut être tracée et justifiée
2. **Hybridité**: Combine le meilleur des règles et du ML
3. **Reproductibilité**: Code open-source, documentation complète
4. **Modularité**: Chaque composant est indépendant et testable

---

Architecture du système



Flux de traitement

1. **Entrée** → URL ou texte brut
2. **Récupération** → Contenu web (si URL)
3. **Prétraitement** → Nettoyage du texte
4. **Données externes** → WHOIS, fact-check APIs
5. **Analyse règles** → Marqueurs linguistiques, réputation
6. **Analyse NLP** → Sentiment, biais, entités
7. **Calcul score** → Pondération hybride (0-1)
8. **Génération rapport** → JSON structuré
9. **Sauvegarde ontologie** → Triplets RDF

Modules et fichiers

Structure du projet

```
syscred/
├── __init__.py              # Package init
├── config.py                # Configuration centralisée
├── verification_system.py   # Système principal
├── api_clients.py           # Clients APIs externes
├── ontology_manager.py      # Gestion OWL/RDF
├── seo_analyzer.py          # Analyse SEO/PageRank
├── backend_app.py           # API Flask REST
├── eval_metrics.py          # Métriques d'évaluation
├── ir_engine.py             # Moteur de recherche
├── requirements.txt         # Dépendances Python
├── setup.py                 # Installation package
├── syscred_kaggle.ipynb     # Notebook Kaggle
├── syscred_colab.ipynb      # Notebook Colab (avec Drive)
└── kaggle_to_gdrive_backup.ipynb  # Backup notebooks
```

Description des modules

`config.py` - Configuration centralisée

But: Centraliser tous les paramètres du système dans un seul fichier.

Classes:

- `Config` - Configuration de base

- `DevelopmentConfig` - Pour développement local
- `ProductionConfig` - Pour production
- `TestingConfig` - Pour tests (ML désactivé)

Paramètres clés:

Paramètre	Description	Valeur par défaut
`HOST`	Adresse du serveur	`0.0.0.0`
`PORT`	Port du serveur	`5000`
`DEBUG`	Mode debug	`true`
`LOAD_ML_MODELS`	Charger les modèles ML	`true`
`WEB_FETCH_TIMEOUT`	Timeout HTTP (sec)	`10`

Pondérations des scores:

```
SCORE_WEIGHTS = {
    'source_reputation': 0.25,  # Réputation de la source
    'domain_age': 0.10,         # Âge du domaine
    'sentiment_neutrality': 0.15, # Neutralité du ton
    'entity_presence': 0.15,    # Présence d'entités vérifiables
    'coherence': 0.15,          # Cohérence textuelle
    'fact_check': 0.20          # Résultats fact-check
}
```

Variables d'environnement:

```
export SYSCRED_ENV=production      # Environnement (dev/prod/testing)
export SYSCRED_PORT=8080           # Port personnalisé
export SYSCRED_GOOGLE_API_KEY=xxx  # Clé Google Fact Check
export SYSCRED_LOAD_ML=false       # Désactiver ML
```

`verification_system.py` - Système principal

But: Pipeline principal de vérification de crédibilité.

Classe principale: `CredibilityVerificationSystem`

Méthodes principales:

Méthode	Description
`__init__()`	Initialise le système, charge les modèles
`verify_information(input)`	Pipeline principal de vérification
`rule_based_analysis(text, data)`	Analyse symbolique
`nlp_analysis(text)`	Analyse NLP (ML)
`calculate_overall_score()`	Calcule le score final
`generate_report()`	Génère le rapport JSON

Modèles ML utilisés:

Modèle	Usage
`distilbert-base-uncased-finetuned-sst-2-english`	Sentiment
`dbmdz/bert-large-cased-finetuned-conll03-english`	NER
`bert-base-uncased`	Détection de biais (placeholder)
`LIME`	Explication des prédictions

`api_clients.py` - Clients APIs externes

But: Abstraire toutes les interactions avec les APIs externes.

Classe principale: `ExternalAPIClients`

APIs intégrées:

API	Méthode	Description
Web Content	`fetch_web_content()`	Récupère et parse le HTML
WHOIS	`whois_lookup()`	Âge et registrar du domaine
Google Fact Check	`google_fact_check()`	Vérification des faits
Source Reputation	`get_source_reputation()`	Base de données interne
CommonCrawl	`estimate_backlinks()`	Estimation backlinks

Data classes:

- `WebContent` - Contenu web parsé
- `DomainInfo` - Informations WHOIS
- `FactCheckResult` - Résultat fact-check
- `ExternalData` - Données agrégées

`ontology_manager.py` - Gestion OWL/RDF

But: Traçabilité sémantique avec ontologie OWL.

Fonctionnalités:

- Chargement d'ontologie de base (.ttl)
- Ajout de triplets RDF pour chaque évaluation
- Sauvegarde des données accumulées
- Requêtes SPARQL

Ontologie utilisée:

- Format: Turtle (.ttl)
- Namespace: `http://syscred.uqam.ca/ontology#`
- Concepts: `Evaluation`, `Source`, `CredibilityScore`, `Evidence`

`backend_app.py` - API Flask

But: Exposer SysCRED via API REST.

Endpoints:

Endpoint	Méthode	Description
`/api/verify`	POST	Vérification principale
`/api/seo`	POST	Analyse SEO uniquement
`/api/ontology/stats`	GET	Statistiques ontologie
`/api/health`	GET	Vérification santé
`/api/config`	GET	Configuration actuelle

Exemple requête:

```bash
curl -X POST http://localhost:5000/api/verify \
  -H "Content-Type: application/json" \
  -d '{"input_data": "https://example.com/article"}'
```

Installation et configuration

Prérequis

- Python 3.8+
- pip
- Git

Installation locale

```bash
# Cloner le repository
git clone https://github.com/DominiqueLoyer/syscred.git
cd syscred

# Créer environnement virtuel
python -m venv venv
source venv/bin/activate  # Linux/Mac
# ou: venv\Scripts\activate  # Windows

# Installer les dépendances
pip install -r requirements.txt

# Installer le package en mode développement
pip install -e .
```

Installation des dépendances

```bash
# Dépendances principales
pip install transformers torch numpy
pip install flask flask-cors
pip install rdflib owlrl
pip install requests beautifulsoup4

# Dépendances optionnelles
pip install python-whois  # Pour WHOIS
pip install lime          # Pour explications ML
```

Fichier requirements.txt

```
transformers>=4.30.0
torch>=2.0.0
numpy>=1.24.0
flask>=2.3.0
flask-cors>=4.0.0
rdflib>=6.3.0
owlrl>=6.0.0
requests>=2.31.0
beautifulsoup4>=4.12.0
python-whois>=0.8.0
lime>=0.2.0
```

Commandes et utilisation

Démarrer l'API Flask

```
# Mode développement
cd /path/to/syscred
python backend_app.py

# Avec variables d'environnement
SYSCRED_PORT=8080 SYSCRED_DEBUG=true python backend_app.py

# Mode production
SYSCRED_ENV=production python backend_app.py
```

Tester le système en ligne de commande

```
# Test direct du module
python -m syscred.verification_system

# Test avec entrée personnalisée
python -c "
from syscred.verification_system import CredibilityVerificationSystem
sys = CredibilityVerificationSystem(load_ml_models=False)
result = sys.verify_information('https://www.lemonde.fr')
print(result['scoreCredibilite'])
"
```

Utilisation dans Kaggle/Colab

Ouvrez le notebook `syscred_kaggle.ipynb` ou `syscred_colab.ipynb` :

```
# Cellule 1: Installation
!pip install transformers torch rdflib requests beautifulsoup4

# Cellule 2: Importer et tester
from syscred import CredibilityVerificationSystem
sys = CredibilityVerificationSystem()
result = sys.verify_information("https://example.com")
```

API REST - Exemples

```
# Vérifier une URL
curl -X POST http://localhost:5000/api/verify \
  -H "Content-Type: application/json" \
  -d '{"input_data": "https://www.bbc.com/article"}'

# Vérifier du texte
curl -X POST http://localhost:5000/api/verify \
  -H "Content-Type: application/json" \
  -d '{"input_data": "This is a verified news report."}'

# Vérifier la santé
curl http://localhost:5000/api/health

# Obtenir la configuration
curl http://localhost:5000/api/config
```

Choix de conception

Pourquoi approche hybride neuro-symbolique?

Approche	Forces	Faiblesses
Règles	Transparent, explicable, rapide	Rigide, couverture limitée
ML/NLP	Flexible, patterns complexes	Boîte noire, besoin données
Hybride	Combine les deux!	Plus complexe

Décision: Utiliser les règles pour les cas clairs (réputation connue, marqueurs linguistiques) et le ML pour les nuances (sentiment, biais).

Pourquoi ces pondérations?

Les poids par défaut reflètent l'importance relative de chaque facteur selon la littérature:

```
SCORE_WEIGHTS = {
    'source_reputation': 0.25,  # Le plus important: source connue
    'fact_check': 0.20,         # Vérification externe
    'sentiment_neutrality': 0.15,
    'entity_presence': 0.15,
    'coherence': 0.15,
    'domain_age': 0.10          # Moins important seul
}
```

Pourquoi LIME pour l'explicabilité?

- **Local Interpretable Model-agnostic Explanations**
- Fonctionne avec n'importe quel modèle
- Génère des explications compréhensibles
- Standard académique reconnu

Pourquoi OWL/RDF?

- **Traçabilité**: Chaque évaluation est enregistrée
- **Raisonnement**: Inférences automatiques possibles (OWL-RL)
- **Interopérabilité**: Standard W3C, compatible SPARQL
- **Publication**: Données linked data

Améliorations réalisées

Version 2.0 (Janvier 2026)

1. **Configuration centralisée** (`config.py`)
   - Variables d'environnement
   - Profils dev/prod/testing
   - Pondérations configurables
2. **API Clients refactorisés** (`api_clients.py`)
   - Data classes typées

- Gestion d'erreurs robuste
- WHOIS lookup réel

3. **Notebooks Kaggle/Colab**
   - `syscred_kaggle.ipynb` - Version Kaggle
   - `syscred_colab.ipynb` - Version avec Google Drive
   - Badges "Open in" pour facilité
4. **Fix du bug `NameError: result`**
   - Variable locale dans section RDF
   - Fallback si aucun résultat
5. **README professionnel**
   - Badge DOI Zenodo
   - Quick start
   - API endpoints documentés
6. **Notebook backup Kaggle→Drive**
   - `kaggle_to_gdrive_backup.ipynb`
   - Sauvegarde automatique

Améliorations futures

Court terme (Prochains mois)

- [ ] **Google Fact Check API réel** - Intégrer la clé API
- [ ] **CommonCrawl backlinks** - Analyse réelle des backlinks
- [ ] **Plus de sources** - Étendre `SOURCE_REPUTATIONS`
- [ ] **Tests unitaires** - Couverture >80%

Moyen terme (6-12 mois)

- [ ] **Modèle de biais fine-tuné** - Entraîner sur données réelles
- [ ] **Cache Redis** - Mise en cache des résultats
- [ ] **Interface web moderne** - React/Vue frontend
- [ ] **Docker** - Conteneurisation

Long terme (Thèse)

- [ ] **Évaluation formelle** - Dataset de benchmark
- [ ] **Multi-langue** - Support français natif
- [ ] **Graphe de connaissances** - Neo4j intégration

- [ ] **Apprentissage continu** - Feedback loop

API Reference

Classe `CredibilityVerificationSystem`

```python
class CredibilityVerificationSystem:
    def __init__(
        self,
        google_api_key: Optional[str] = None,
        ontology_base_path: Optional[str] = None,
        ontology_data_path: Optional[str] = None,
        load_ml_models: bool = True
    ):
        """
        Initialize the credibility verification system.

        Args:
            google_api_key: API key for Google Fact Check
            ontology_base_path: Path to base ontology TTL
            ontology_data_path: Path to store data
            load_ml_models: Whether to load ML models
        """

    def verify_information(self, input_data: str) -> Dict[str, Any]:
        """
        Main pipeline to verify credibility.

        Args:
            input_data: URL or text to verify

        Returns:
            Complete evaluation report with:
            - idRapport: Unique report ID
            - scoreCredibilite: 0.0-1.0
            - resumeAnalyse: French summary
            - detailsScore: Score breakdown
            - reglesAppliquees: Rule-based results
            - analyseNLP: NLP analysis results
        """
```

Classe `Config`

```python
class Config:
    # Chemins
    BASE_DIR: Path
    ONTOLOGY_BASE_PATH: Path
    ONTOLOGY_DATA_PATH: Path

    # Serveur
    HOST: str = "0.0.0.0"
    PORT: int = 5000
    DEBUG: bool = True

    # API Keys
    GOOGLE_FACT_CHECK_API_KEY: Optional[str]

    # Modèles ML
    LOAD_ML_MODELS: bool = True
    SENTIMENT_MODEL: str
    NER_MODEL: str

    # Pondérations
    SCORE_WEIGHTS: Dict[str, float]
    CREDIBILITY_THRESHOLDS: Dict[str, float]
    SOURCE_REPUTATIONS: Dict[str, str]

    @classmethod
    def load_external_reputations(cls, filepath: str) -> None:
        """Charger réputations depuis fichier JSON."""

    @classmethod
    def update_weights(cls, new_weights: Dict[str, float]) -> None:
        """Mettre à jour les pondérations."""

    @classmethod
    def to_dict(cls) -> Dict:
        """Exporter configuration en dictionnaire."""
```

Ontologie OWL

Structure conceptuelle

```
syscred:Evaluation
  └── syscred:evaluates → syscred:Information
```

```
  └── syscred:hasScore → xsd:float
  └── syscred:hasEvidence → syscred:Evidence
  └── syscred:generatedAt → xsd:dateTime

syscred:Information
  └── syscred:hasSource → syscred:Source
  └── syscred:hasContent → xsd:string

syscred:Source
  └── syscred:hasDomain → xsd:string
  └── syscred:hasReputation → syscred:ReputationLevel
  └── syscred:hasDomainAge → xsd:integer

syscred:Evidence
  └── syscred:type → xsd:string (Linguistic, FactCheck, etc.)
  └── syscred:value → xsd:string
  └── syscred:impact → xsd:float
```

Exemple de triplets générés

```turtle
@prefix syscred: <http://syscred.uqam.ca/ontology#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

syscred:eval_1705890000 a syscred:Evaluation ;
    syscred:evaluates syscred:info_lemonde_article ;
    syscred:hasScore "0.85"^^xsd:float ;
    syscred:generatedAt "2026-01-21T13:40:00"^^xsd:dateTime ;
    syscred:hasEvidence syscred:evidence_1 .

syscred:evidence_1 a syscred:Evidence ;
    syscred:type "SourceReputation" ;
    syscred:value "High" ;
    syscred:impact "0.25"^^xsd:float .
```

Scripts utilitaires

Script de backup vers Obsidian/Notion

Créez ce script dans `/Users/bk280625/documents041025/MonCode/` :

```bash
#!/bin/bash
# save_syscred_docs.sh
```

```
# Usage: ./save_syscred_docs.sh

DOC_SOURCE="/Users/bk280625/documents041025/MonCode/syscred/SysCRED_Document
ation.md"
OBSIDIAN_VAULT="/Users/bk280625/Documents/Obsidian/PhD"
DATE=$(date +%Y%m%d)

# Copier vers Obsidian
cp "$DOC_SOURCE" "$OBSIDIAN_VAULT/SysCRED_Documentation_$DATE.md"
echo "✅ Copié vers Obsidian: $OBSIDIAN_VAULT"

# Ouvrir dans Obsidian (Mac)
open "obsidian://open?vault=PhD&file=SysCRED_Documentation_$DATE"

# Pour Notion: utiliser l'API ou copier manuellement
# Notion n'a pas d'import direct de fichiers locaux
echo "📋 Pour Notion: Copiez le contenu de $DOC_SOURCE"
echo "   Ou utilisez: https://notion.so/import"
```

Références

- Loyer, D. S. (2025). *Modeling and Hybrid System for Verification of Sources Credibility*. UQAM.
- Loyer, D. S. (2025). *Ontology of a Verification System for Liability of the Information*. DIC-9335.

Documentation générée le 21 janvier 2026
SysCRED v2.0 - Dominique S. Loyer - UQAM