Pilier 3 AMÉLIORÉ : LLM as a Judge - Évaluation de Qualité

Objectif

Évaluer la qualité des analyses produites par les deux LLM précédents (Pertinence Checker et Risk Analyzer) avec un système de scoring avancé, adaptatif et explicable.

Processus d'Évaluation

Le LLM Judge évalue **deux niveaux** :

1. **Évaluation du Pertinence Checker (Agent 1B)** : La conclusion de pertinence est-elle correcte et bien justifiée ?
2. **Évaluation du Risk Analyzer (Agent 2)** : Les impacts et recommandations sont-ils pertinents, complets et appropriés ?

Critères d'Évaluation (8 critères au total)

Critères de Base (6 critères originaux)

Critère	Description	Applicable à
Source Relevance	La source citée est-elle fiable et pertinente ?	Agent 1B + Agent 2
Company Data Alignment	Les données internes sont-elles correctement interprétées ?	Agent 1B + Agent 2
Logical Coherence	La conclusion découle-t-elle logiquement des preuves ?	Agent 1B + Agent 2
Completeness	L'analyse couvre-t-elle tous les aspects importants ?	Agent 2 uniquement
Recommendation Appropriateness	Les recommandations sont-elles concrètes et applicables ?	Agent 2 uniquement
Traceability	Chaque affirmation est-elle tracée à une source ou une	Agent 1B + Agent 2

	donnée ?	

Critères Additionnels Hutchinson (2 nouveaux critères)

Critère	Description	Applicable à
Strategic Alignment	L'analyse est-elle alignée avec les priorités stratégiques d'Hutchinson ?	Agent 2 uniquement
Actionability Timeline	Les recommandations ont-elles des timelines réalistes et actionnables ?	Agent 2 uniquement

---

Scoring Pondéré par Type de Risque

Pondération Climatique

Python

```
weights_climatique = {
    "source_relevance": 1.0,
    "company_data_alignment": 1.5,      # Plus important (géolocalisation)
    "logical_coherence": 1.0,
    "completeness": 1.0,
    "recommendation_appropriateness": 1.2,
    "traceability": 1.5,                # Critique (distance GPS)
    "strategic_alignment": 1.0,
    "actionability_timeline": 1.3       # Urgence temporelle
}
```

Pondération Réglementaire

Python

```
weights_reglementaire = {
    "source_relevance": 2.0,            # CRITIQUE (source légale)
    "company_data_alignment": 1.0,
    "logical_coherence": 1.2,
    "completeness": 1.3,                # Tous les aspects légaux
```

```
    "recommendation_appropriateness": 1.0,
    "traceability": 1.8,                # CRITIQUE (références légales)
    "strategic_alignment": 1.0,
    "actionability_timeline": 1.0
}
```

Pondération Géopolitique

Python

```
weights_geopolitique = {
    "source_relevance": 1.5,
    "company_data_alignment": 1.2,
    "logical_coherence": 1.3,           # Analyse complexe
    "completeness": 1.4,                # Cascade multi-niveaux
    "recommendation_appropriateness": 1.5, # Actions stratégiques
    "traceability": 1.2,
    "strategic_alignment": 1.6,         # CRITIQUE (impact stratégique)
    "actionability_timeline": 1.4       # Urgence variable
}
```

Calcul du Score Pondéré

Python

```
def calculate_weighted_score(scores: Dict[str, float], weights: Dict[str, float
    """
    Calcule le score global pondéré

    scores: {criterion: score (0-10)}
    weights: {criterion: weight (0.5-2.0)}

    Returns: score pondéré (0-10)
    """
    total_weighted = sum(scores[c] * weights[c] for c in scores)
    total_weight = sum(weights.values())
    return total_weighted / total_weight
```

Structure de Sortie JSON AMÉLIORÉE

JSON

```
{
  "event_id": "EVT-2024-001",
  "event_type": "climatique",
  "judge_evaluation": {
    "pertinence_checker_evaluation": {
      "source_relevance": {
        "score": 9,
        "confidence": 0.95,
        "comment": "Source officielle et directe",
        "evidence": [
          "Document publié par l'agence météorologique nationale",
          "Coordonnées GPS précises fournies",
          "Historique de fiabilité de la source: 98%"
        ],
        "weaknesses": []
      },
      "company_data_alignment": {
        "score": 8,
        "confidence": 0.88,
        "comment": "Données correctement interprétées",
        "evidence": [
          "Site Bangkok identifié à 0 km de l'événement",
          "Fournisseur Thai Rubber à 11.89 km",
          "Correspondance avec la base de données interne"
        ],
        "weaknesses": [
          "Pourrait vérifier les sites secondaires dans un rayon de 100 km"
        ]
      },
      "logical_coherence": {
        "score": 9,
        "confidence": 0.92,
        "comment": "Conclusion bien justifiée",
        "evidence": [
          "Raisonnement clair: proximité géographique → impact direct",
          "Prise en compte du rayon d'impact (50 km)",
          "Cohérence avec les événements similaires passés"
        ],
        "weaknesses": []
      },
      "traceability": {
        "score": 10,
        "confidence": 0.98,
        "comment": "Traçabilité complète",
        "evidence": [
          "Chaque entité impactée est liée à une distance GPS",
          "Source de l'événement clairement identifiée",
```

```
          "Références aux données internes (site_bangkok, supplier_thai_rubber)
        ],
        "weaknesses": []
      },
      "weighted_score": 9.1,
      "confidence_overall": 0.93
    },
    "risk_analyzer_evaluation": {
      "source_relevance": {
        "score": 8,
        "confidence": 0.85,
        "comment": "Passages sources bien sélectionnés",
        "evidence": [
          "Utilisation des données de l'événement (rayon, durée)",
          "Référence aux données fournisseur (stock, délais)",
          "Contexte historique des inondations à Bangkok"
        ],
        "weaknesses": [
          "Pourrait inclure des données météorologiques historiques"
        ]
      },
      "company_data_alignment": {
        "score": 9,
        "confidence": 0.90,
        "comment": "Impacts correctement identifiés",
        "evidence": [
          "Criticité fournisseur unique détectée",
          "Stock vs durée calculé (14 jours < 21 jours)",
          "Cascade d'impacts identifiée (Bangkok → Europe)"
        ],
        "weaknesses": []
      },
      "logical_coherence": {
        "score": 8,
        "confidence": 0.87,
        "comment": "Recommandations cohérentes avec l'analyse",
        "evidence": [
          "Activation backup Vietnam (car double-source disponible)",
          "Transport aérien d'urgence (car stock insuffisant)",
          "Timeline cohérente avec la durée de l'événement"
        ],
        "weaknesses": [
          "Pourrait quantifier le coût du transport aérien vs normal"
        ]
      },
      "completeness": {
        "score": 7,
        "confidence": 0.82,
```

```
      "comment": "Analyse complète mais pourrait être enrichie",
      "evidence": [
        "Sites impactés: identifiés",
        "Fournisseurs impactés: identifiés",
        "Cascade d'impacts: analysée",
        "Recommandations: 9 actions priorisées"
      ],
      "weaknesses": [
        "Pourrait inclure une analyse financière détaillée",
        "Impact sur les clients finaux non quantifié",
        "Pas d'analyse de scénarios alternatifs"
      ]
    },
    "recommendation_appropriateness": {
      "score": 9,
      "confidence": 0.91,
      "comment": "Recommandations concrètes et actionnables",
      "evidence": [
        "Actions IMMEDIATE: Activer backup Vietnam (24-48h)",
        "Actions HIGH: Transport aérien d'urgence (48-72h)",
        "Actions MEDIUM: Communication clients (3-7 jours)",
        "Coûts estimés fournis (50K EUR, 120K EUR...)",
        "Responsables identifiés (Supply Chain Manager, Procurement)"
      ],
      "weaknesses": []
    },
    "traceability": {
      "score": 9,
      "confidence": 0.89,
      "comment": "Traçabilité quasi-complète",
      "evidence": [
        "Chaque impact lié à une entité (site/fournisseur)",
        "Chaque recommandation justifiée par l'analyse",
        "Références aux données internes (stock, délais, backup)"
      ],
      "weaknesses": [
        "Certaines estimations financières non tracées à une source"
      ]
    },
    "strategic_alignment": {
      "score": 8,
      "confidence": 0.86,
      "comment": "Aligné avec les priorités Hutchinson",
      "evidence": [
        "Priorisation de la continuité de production",
        "Protection des sites stratégiques (Bangkok = fort)",
        "Gestion proactive des risques supply chain",
        "Communication transparente avec les clients"
```

```
        ],
        "weaknesses": [
          "Pourrait mentionner l'impact sur les objectifs ESG"
        ]
      },
      "actionability_timeline": {
        "score": 9,
        "confidence": 0.90,
        "comment": "Timelines réalistes et bien structurées",
        "evidence": [
          "Actions IMMEDIATE: 0-24h (alerte interne)",
          "Actions HIGH: 24-72h (activation backup, transport)",
          "Actions MEDIUM: 3-7 jours (communication, monitoring)",
          "Actions LOW: 1-6 mois (diversification, contrats)",
          "Timeline cohérente avec l'urgence (stock 14 jours)"
        ],
        "weaknesses": []
      },
      "weighted_score": 8.4,
      "confidence_overall": 0.88
    },
    "overall_quality_score": 8.7,
    "overall_confidence": 0.90,
    "action_recommended": "APPROVE",
    "reasoning": "Score global de 8.7 (> 8.5) avec confiance élevée (0.90). L'a
    "metadata": {
      "judge_model": "claude-sonnet-4-5-20250929",
      "evaluation_timestamp": "2026-01-31T14:23:45Z",
      "weights_used": "climatique",
      "total_criteria_evaluated": 12
    }
  }
}
```


Seuils de Décision AMÉLIORÉS

Score Global	Confiance	Action	Justification
⩾ 8.5	⩾ 0.85	**APPROVE**	Alerte immédiate - Haute qualité et haute confiance
⩾ 8.5	< 0.85	**REVIEW**	Validation humaine - Score élevé mais confiance faible

7.0 - 8.4	⩾ 0.80	**REVIEW**	Validation humaine - Qualité acceptable
7.0 - 8.4	< 0.80	**REVIEW_PRIORITY**	Validation humaine prioritaire - Confiance faible
< 7.0	-	**REJECT**	Archiver - Qualité insuffisante

Feedback Loop avec Ground Truth

Processus d'Amélioration Continue

Python

```
class JudgeFeedbackLoop:
    """
    Système d'amélioration continue du Judge basé sur les validations humaines
    """

    def __init__(self):
        self.disagreements = []
        self.ground_truth_cases = []

    def log_disagreement(self, case_id: str, judge_decision: str,
                         human_decision: str, human_reasoning: str):
        """
        Enregistre un désaccord entre le Judge et l'humain
        """
        self.disagreements.append({
            "case_id": case_id,
            "judge_decision": judge_decision,
            "human_decision": human_decision,
            "human_reasoning": human_reasoning,
            "timestamp": datetime.now()
        })

        # Tous les 10 désaccords, analyser et ajuster
        if len(self.disagreements) % 10 == 0:
            self.analyze_and_adjust()

    def analyze_and_adjust(self):
        """
```

```
        Analyse les désaccords et ajuste les prompts du Judge
        """
        # Identifier les patterns de désaccords
        patterns = self._identify_patterns(self.disagreements[-10:])

        # Ajuster les prompts
        for pattern in patterns:
            if pattern["type"] == "over_optimistic":
                # Le Judge est trop optimiste
                self._adjust_threshold(pattern["criterion"], direction="stricte
            elif pattern["type"] == "under_optimistic":
                # Le Judge est trop strict
                self._adjust_threshold(pattern["criterion"], direction="lenient

    def calculate_judge_accuracy(self) -> float:
        """
        Calcule la précision du Judge par rapport aux validations humaines
        """
        if not self.disagreements:
            return 1.0

        total_cases = len(self.disagreements) + len(self.ground_truth_cases)
        correct_decisions = total_cases - len(self.disagreements)

        return correct_decisions / total_cases
```

Métriques de Performance du Judge

Métrique	Calcul	Cible
Judge Accuracy	% de cas où le Judge et l'humain sont d'accord	⩾ 92%
False Approve Rate	% de cas APPROVE rejetés par l'humain	⩽ 5%
False Reject Rate	% de cas REJECT approuvés par l'humain	⩽ 3%
Review Efficiency	% de cas REVIEW qui nécessitent vraiment une validation	⩾ 85%

Workflow Complet de Qualité AMÉLIORÉ

Plain Text

```
1. Événement Externe
    ↓
2. LLM Pertinence Checker (Agent 1B)
    ├─ NON (< 0.5) → Archiver
    ├─ PARTIELLEMENT (0.5-0.7) → Review Humaine Rapide
    └─ OUI (> 0.7) ↓
3. LLM Risk Analyzer (Agent 2)
    ↓
4. LLM Judge (Agent 3) - Évaluation avec scoring pondéré
    ├─ Score ≥ 8.5 + Confiance ≥ 0.85 → APPROVE → Alerte Immédiate
    ├─ Score ≥ 8.5 + Confiance < 0.85 → REVIEW
    ├─ Score 7.0-8.4 → REVIEW
    └─ Score < 7.0 → REJECT → Archiver
    ↓
5. Validation Humaine (si REVIEW)
    ├─ Approuvé → Alerte + Log Feedback
    ├─ Modifié → Alerte avec modifications + Log Feedback
    └─ Rejeté → Archiver + Log Feedback
    ↓
6. Feedback Loop
    └─ Ajustement des prompts tous les 10 cas
    ↓
7. Ground Truth Validation (Continu)
    └─ Mesurer la fiabilité globale (Cible: ≥ 90%)
```

Avantages des Améliorations

✅ **Scoring Adaptatif** : Les critères sont pondérés selon le type de risque (climatique, réglementaire, géopolitique)

✅ **Confiance Explicite** : Chaque score a un niveau de confiance, permettant de détecter les cas incertains

✅ **Explainability Renforcée** : Evidence et weaknesses pour chaque critère, facilitant la validation humaine

✅ **Critères Hutchinson** : Strategic Alignment et Actionability Timeline ajoutés pour mieux refléter les besoins métier

✅ **Feedback Loop** : Le système s'améliore continuellement grâce aux validations humaines

✅ **Métriques de Performance** : Suivi de la précision du Judge (cible: ⩾ 92%)

✅ **Décision Nuancée** : Prise en compte de la confiance en plus du score pour la décision finale

---

Implémentation Technique

Fichiers à Créer

1. `agent_3/judge.py` : Agent Judge principal avec scoring pondéré
2. `agent_3/criteria_evaluator.py` : Évaluation des 8 critères avec confiance
3. `agent_3/weights_config.py` : Configuration des poids par type de risque
4. `agent_3/prompts.py` : Prompts structurés pour le LLM
5. `agent_3/feedback_loop.py` : Système d'amélioration continue
6. `agent_3/test_judge.py` : Tests avec les résultats d'Agent 2
7. `agent_3/__init__.py` : Exports

Technologies

- **LLM** : Claude Sonnet 4.5 (préférence utilisateur)
- **Framework** : LangGraph pour orchestration
- **Base de données** : Table `judge_evaluations` dans SQLite/MySQL
- **Métriques** : Table `judge_performance_metrics`

Prochaines Étapes

1. ✅ Valider cette spécification améliorée avec vous
2. ✅ Implémenter Agent 3 avec toutes les améliorations
3. ✅ Créer les tests avec les résultats d'Agent 2
4. ✅ Intégrer dans le workflow LangGraph complet
5. ✅ Déployer et monitorer les performances

Voulez-vous que je commence l'implémentation maintenant ? 🚀